IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
MARSHALL DIVISION

NOKIA TECHNOLOGIES OY AND ALCATEL-LUCENT USA INC. **Plaintiffs,** **v.** **APPLE INC.,** **Defendants.**	**Civil Action No. 2:16-cv-1440** **JURY TRIAL REQUESTED**

ORIGINAL COMPLAINT

Plaintiffs Nokia Technologies Oy ("Nokia Tech") and Alcatel-Lucent USA Inc. ("ALU") (together, "Nokia") file this Original Complaint against Apple Inc. ("Apple") and allege as follows:

NATURE OF THE ACTION

1. Each of the Nokia Patents-in-Suit (defined below) relates to video coding.[1] For example, each of the Nokia Patents-in-Suit include claims directed to encoding or decoding video compliant with the H.264 Advanced Video Coding standard promulgated by the International Telecommunication Union ("ITU"). Apple's products, which support H.264 video, including the iPhone, iPad, iPod, Apple Watch, Mac computer products, and Apple digital media players such as Apple TV, infringe Nokia's patents asserted in this case. Nokia's patents asserted in this suit cover fundamental and innovative contributions made by Nokia to video coding

[1] All descriptions of the inventions herein are presented to give a general background of those inventions. Such statements are not intended to be used, nor should be used, for purposes of patent claim interpretation. Nokia presents these statements subject to, and without waiver of, its right to argue that claim terms should be construed in a particular way, as contemplated by claim interpretation jurisprudence and the relevant evidence

technologies, including the H.264 standard, that reduce the amount of digital data needed to represent video. Nokia's innovations allow video to be transmitted over communications networks, such as cellular networks, with high quality and dramatically lower bandwidth requirements. Nokia's innovations also significantly reduce the size of video files, allowing more efficient storage on mobile devices such as the iPhone and iPad. Apple has benefitted greatly from Nokia's innovations, which have enabled Apple products to more efficiently and effectively stream video over communications networks.

2. Despite all the advantages that have been enjoyed by Apple, Apple has steadfastly refused to agree to license Nokia's H.264 patents on reasonable terms. Dozens of companies have licensed Nokia's patents for use in their products that support the H.264 standard at Nokia's established royalty rates. Apple, however, refuses to pay Nokia's established royalty rates. Apple's unwillingness to negotiate a license to Nokia's patented H.264 technology in good faith has forced Nokia to institute this case for patent infringement.

PARTIES

3. Plaintiff Nokia Tech is a Finnish corporation with its principal place of business at Karaportti 3, FIN-02610, Espoo, Finland**.**

4. Nokia Tech is a wholly-owned subsidiary of Nokia Corporation ("Nokia Corp.") and is the sole owner by assignment of all right, title, and interest in certain of the Patents-in-Suit.

5. Plaintiff ALU is a Delaware corporation with its principal places of business at 600 Mountain Avenue, Murray Hill, NJ 07974 and 601 Data Drive, Plano, Texas 75075. ALU conducts significant business operations at its principal place of business at 601 Data Drive, Plano, Texas 75075.

6. ALU is a wholly-owned indirect subsidiary of Alcatel-Lucent S.A. ("Alcatel-Lucent"). In January 2016, Nokia Corp. obtained a controlling interest in Alcatel-Lucent S.A. and ALU became an affiliate of Nokia Corp. ALU is the sole owner by assignment of all right, title and interest in certain of the Patents-in-Suit.

7. Defendant Apple Inc. is a California corporation with its principal place of business at 1 Infinite Loop, M/S 38-3TX, Cupertino, California 95014. Apple designs, manufactures, uses, imports into the United States, sells, and/or offers for sale in the United States smartphones, tablets, computers, and other products and services that practice the H.264 Standard. Apple's devices are marketed, offered for sale, and/or sold throughout the United States, including within this District.

JURISDICTION AND VENUE

8. This Court has exclusive subject matter jurisdiction over this case under 28 U.S.C. §§ 1331 and 1338.

9. Venue is proper in this Court pursuant to 28 U.S.C. §§1391 and 1400(b).

10. This Court has personal jurisdiction over Apple. Apple has continuous and systematic business contacts with the State of Texas. Apple, directly or through subsidiaries or intermediaries (including distributors, retailers, and others), conducts its business extensively throughout Texas, by shipping, distributing, offering for sale, selling, and advertising (including the provision of interactive web pages) its products and services in the State of Texas and the Eastern District of Texas. Apple, directly and through subsidiaries or intermediaries (including distributors, retailers, and others), has purposefully and voluntarily placed its infringing products and services into this District and into the stream of commerce with the intention and expectation that they will be purchased and used by consumers in this District. These infringing products and

services have been and continue to be purchased and used by consumers in this District. Apple has committed acts of patent infringement within the State of Texas and, more particularly, within the Eastern District of Texas. Jurisdiction over Apple in the matter is also proper inasmuch as Apple has voluntarily submitted itself to the jurisdiction of the courts by commencing litigations within the State of Texas, by registering with the Texas Secretary of State's Office to do business in the State of Texas, and by appointing a registered agent.

NOKIA'S COMPLIANCE WITH RAND

A. The ITU and H.264 Standardization Process.

11. In order to enable devices manufactured by different entities to interoperate, standards-setting organizations ("SSOs") have formed to promulgate industry standards, which define communication protocols that can be adopted by different manufacturers to enable interoperability between their devices.

12. The International Telecommunications Union (ITU) and the International Standards Organization (ISO) jointly publish a standard that is referred to as "H.264," "MPEG-4 part 10," or "Advanced Video Coding." The H.264 Standard development process was initiated by the Video Coding Experts Group ("VCEG") and finalized by the Joint Video Team ("JVT"), which was a collaborative effort between VCEG and the Moving Picture Experts Group ("MPEG").

13. The ITU was formed in 1865 at the International Telegraph Convention and, in 1947, became a specialized agency of the United Nations, responsible for issues that concern information and communication technologies. The ITU handles a number of different matters and thus is organized into various sectors. One of the sectors is Telecommunication Standardization or "ITU-T." The mission of ITU-T is to ensure efficient and timely production

of standards related to the field of telecommunications. The standards developed by ITU-T are referred to as "Recommendations."

14. Within ITU-T, members come together in various teams or groups to propose technology that best meets the aims of the organization and its members and draft the Recommendations. The goal is to draft Recommendations that incorporate the best available technology to ensure that the standards are the highest possible quality. The H.264 Standard described above is explicitly detailed in the H.264 Recommendation.

15. In order to facilitate the widespread adoption of its standards, the ITU must ensure both that manufacturers have access to the standards and simultaneously ensure that patent holders have incentives to continue to innovate and contribute technologies to the standards. Without the former, standards could issue, but no one would be able to adopt them. Without the latter, there likely would be no standards at all, and manufacturers would be forced to rely on a multiplicity of proprietary technologies.

16. As it searches for the best available technical solutions, the ITU takes into account that many parts of its standards will be covered by patents owned by members and third parties. In order to assist with the usage of patented technologies in standardized communication protocols, the ITU adopted a Common Patent Policy which states that "a patent embodied fully or partly in a Recommendation | Deliverable must be accessible to everybody without undue constraints." This patent policy applies to the ITU, ITU-T, ISO, and IEC.

17. The ITU published Guidelines for compliance with the implementation of the Common Patent Policy, which explain that the Common Patent Policy "was drafted in its operative part as a checklist, covering the three different cases which may arise if a

Recommendation | Deliverable requires licenses for Patents to be practiced or implemented, fully or partly" (Ex. 1 ("ITU Patent Guidelines") at 1).

18. The Common Patent Policy states:

> 2. If a Recommendation | Deliverable is developed and such information as referred to in paragraph 1 has been disclosed, three different situations may arise:
>
> 2.1 The patent holder is willing to negotiate licences free of charge with other parties on a non-discriminatory basis on reasonable terms and conditions. Such negotiations are left to the parties concerned and are performed outside ITU-T/ITU-R/ISO/IEC.
>
> 2.2 The patent holder is willing to negotiate licences with other parties on a non-discriminatory basis on reasonable terms and conditions. Such negotiations are left to the parties concerned and are performed outside ITU-T/ITU-R/ISO/IEC.
>
> 2.3 The patent holder is not willing to comply with the provisions of either paragraph 2.1 or paragraph 2.2; in such case, the Recommendation | Deliverable shall not include provisions depending on the patent.
>
> 3. Whatever case applies (2.1, 2.2 or 2.3), the patent holder has to provide a written statement to be filed at ITU-TSB, ITU-BR or the offices of the CEOs of ISO or IEC, respectively, using the appropriate "Patent Statement and Licensing Declaration" form. This statement must not include additional provisions, conditions, or any other exclusion clauses in excess of what is provided for each case in the corresponding boxes of the form.

(Ex. 2 ("ITU Common Patent Policy") at 1).

19. The Guidelines define the term "Patent" to be "those claims contained in and identified by patents, utility models and other similar statutory rights based on inventions (including applications for any of these) solely to the extent that any such claims are essential to the implementation of a Recommendation | Deliverable. Essential patents are patents that would be required to implement a specific Recommendation | Deliverable" (ITU Patent Guidelines at 2). The definition of "Patent" provided by the Guidelines is mirrored in the Patent Statement and Licensing Declaration Form, which is prepared by each patent holder when declaring patents as

essential to H.264. The ITU thus deems "essential" only patent claims that are required for implementation of a specific Recommendation.

20. The H.264 Recommendation specifies the implementation of decoders and specifically defines the "decoding process" as "[t]he process specified in this Recommendation | International Standard that reads a bitstream and derives decoded pictures from it." *See* ITU-T Rec. H.264, "Advanced video coding for generic audiovisual services" (03/2005) (available at https://www.itu.int/rec/T-REC-H.264-200503-S/en, last visited December 13, 2016) ("H.264 Standard") at 6). It does not, however, specify the implementation of encoders. In fact, it specifically defines "encoder" as "an embodiment of an encoding process," and then defines "encoding process" as "a process, *not specified* in this Recommendation | International Standard, that produces a bitstream conforming to this Recommendation | International Standard." *Id*. at 6 (emphasis added)). As a result, since encoder implementations are not specified under the H.264 Standard, claims covering such encoders are not essential under the Common Patent Policy, and thus any such claims are not subject to a RAND commitment under that Policy.

B. Nokia's Compliance with ITU Licensing Obligations

21. Consistent with the Common Patent Policy, Nokia has declared to ITU that it is prepared to grant licenses to the essential claims of the Asserted Patents on reasonable and non-discriminatory ("RAND") terms and conditions. Specifically, claims of the Asserted Patents directed to the decoder specified in the H.264 Standard are essential, and Nokia has been and remains prepared to grant a license under RAND terms to any essential claims of the Asserted Patents.

22. Nokia is not under a RAND obligation for any claims of the Asserted Patents directed to an encoder or a method of using an encoder because the operation of the encoder is

not specified by the H.264 Standard. Instead, the H.264 Standard specifies only the decoder operation, which is implemented separately from the encoder.

23. Although not obligated to offer a license to non-essential encoder claims, Nokia has nonetheless offered a license to these claims to Apple on terms that would be RAND-compliant if such obligation applied. Nokia has therefore gone beyond what its RAND obligations require.

24. Nokia has negotiated in good faith and made substantial efforts to enter into a license agreement with Apple on reasonable and non-discriminatory terms, as proven by Nokia's conduct in licensing negotiations with Apple.

25. The parties have been discussing a license to Nokia's H.264 essential patents for over two years. For example, on August 31, 2014, Nokia notified Apple that it infringed more than 30 Nokia patents for decoding and encoding video, including certain of the Asserted Patents. At that time, Nokia requested that Apple enter into a license under Nokia's industry-established royalty rates.

26. On November 4, 2014, Nokia and Apple met in Sunnyvale, California, and Nokia gave Apple a presentation in which it identified patent numbers, claims, and corresponding H.264 Standard sections and explained in detail the basis upon which Apple's products infringed certain Nokia video compression patents. The purpose of this meeting was to help Apple assess the technical merits of Nokia's claims and confirm the value of Nokia's H.264 patents.

27. On May 12, 2015, Nokia provided Apple with additional information regarding its licensing program, including the number of licensees and a description of the broad array of licensed technologies. At this meeting, Nokia specifically provided Apple with license offers to its video compression patent portfolio for H.264 decoding, to its video compression patent

portfolio for H.264 encoding, and to a combination of both portfolios. These license offers included reasonable and non-discriminatory rates for the H.264-essential patents and, although not required by ITU policy, reasonable and non-discriminatory rates for Nokia's video encoding patents. Apple rejected these offers.

28. Nokia's May 12, 2015 license offer included Nokia's standard licensing terms. These terms are consistent with rates offered to Nokia's other licensees for the same technology, and many other companies have accepted the same rates.

29. In letters dated May 29, 2015 and June 17, 2015, Nokia provided Apple with 31 additional claim charts regarding video compression patents relevant to its license offers of May 12, 2015.

30. On June 24, 2015, Nokia invited Apple to send technical experts to Nokia's Texas offices in order to discuss any remaining technical questions regarding Nokia's infringement claims. Prior to the meeting, Nokia provided eight additional claim charts to Apple and, on September 9, 2015, sent a summary letter identifying all claim charts that had previously been provided. The technical meeting itself was held on September 22, 2015, and Nokia addressed all specific technical questions raised by Apple during that meeting.

31. During the course of this and previous meetings, Apple failed to advance any meritorious non-infringement arguments. Accordingly, Nokia fully responded to Apple's inquiries and provided Apple detailed information regarding Nokia's claims of infringement, including identifying the relevant portions of its portfolio essential to the H.264 Standard and specifying the ways that the patents have been infringed. Moreover, Apple has been fully aware of Nokia's claims of infringement since, at the very latest, the technical meeting on September 22, 2015, which occurred over a year ago.

32. Apple and Nokia had additional meetings on October 30, 2015 and December 8, 2015, in which Nokia provided further explanations of its RAND license offers. Nokia informed Apple that more licensees to Nokia's H.264 portfolio are being added every month, and that about two-thirds of its licensees at the time were for standard-essential decoding technology, with the balance of licensees electing to license patents for encoding H.264-compliant video or for both encoding and decoding.

33. In a letter dated March 17, 2016, Nokia explained in detail the reasoning underlying the rates of its prior license offers to Apple and simultaneously provided another revised license offer to Apple together with an offer to arbitrate any disputed terms of the offer. Nokia requested a response from Apple within 60 days.

34. Apple did not respond to this revised offer within the time allotted. Consequently, Nokia informed Apple on May 23, 2016 that it considered the offers rejected. Apple's refusal to even react to Nokia's offer to arbitrate in the reasonable timeframe requested by Nokia strongly confirms Apple's unwillingness to take a RAND license and bluntly contradicts its general averments that it honors and respects Nokia's intellectual property. Nokia answered additional technical questions at a meeting in Sunnyvale, California on July 28, 2016.

35. On September 14, 2016, Nokia responded to other communications from Apple and indicated that it remains willing to have any open licensing issues finally resolved in arbitration. Apple has thus far rejected all proposals made by Nokia to engage in binding arbitration to determine a RAND rate for a license under Nokia's H.264 portfolio. Nokia has continuously negotiated with Apple in good faith in order to license its essential patents on RAND terms and has maintained an open dialogue with Apple and met with Apple on many occasions.

36. Nokia has now concluded at least 48 video coding licenses. All licenses that have been concluded since February 2014 are portfolio licenses covering Nokia's H.264 portfolio. Twenty-eight of those licenses relate to decoding, sixteen relate to encoding, and four to the combination of decoding and encoding. Nokia's licenses relate to a vast array of product types, including mobile devices, tablets, televisions, Blu-ray players, cameras, set-top boxes, and video playback devices. Due to the wide range of applications of H.264, Nokia's existing licensees include companies of various sizes that are active in a variety of businesses and industries, with a focus on producers of computer, video playback and entertainment devices (e.g., screens, monitors, set-top boxes, TVs, and Blu-ray players), and camera systems (e.g., general video cameras and special purpose video cameras such as security, door and medical cameras).

37. Nokia's license offers to Apple were made using Nokia's standard licensing terms that are consistent with rates offered to Nokia's other licensees for the same technology, and which other companies have accepted. Nokia has been transparent about its pricing expectations, explained many times its methods for deriving RAND license terms and rates, supported its offers by supplying claim charts detailing how Apple products practice many of Nokia's standards essential patent claims, and has even offered to make licensing concessions during the negotiations.

38. Despite Nokia's good faith efforts to license its H.264 standard-essential patents on RAND terms, Apple has never seriously engaged Nokia and has instead engaged in delay tactics. Apple has persistently refused to constructively negotiate a license, engaged in hold out, and refused Nokia's proposals to allow an independent arbitrator to resolve the parties' disputes regarding an appropriate RAND rate and appropriate terms and conditions.

39. Apple's conduct during these negotiations has proven that it is unwilling to take a license to Nokia's essential patents on RAND terms and that Apple's only interest is in further delay and free riding. For example, Apple has:

i. Repeatedly engaged in dilatory license tactics while making use of Nokia's intellectual property for years;

ii. Refused to accept Nokia's RAND license offers;

iii. Failed to pay compensation of any kind for use of Nokia's H.264 patents;

iv. Failed to make a RAND license offer; and

v. Refused Nokia's proposals to engage in binding arbitration to settle any disputed terms of a RAND license agreement, a process which would unquestionably result in a license on RAND terms.

40. Nokia has complied with all conditions precedent under its RAND commitment to seek the relief it requests in this case. Under such circumstances, this Complaint is necessary to put an end to Apple's conduct as an unwilling licensee.

THE NOKIA PATENTS

41. On May 12, 2009, the U.S. Patent and Trademark Office duly and legally issued U.S. Patent No. 7,532,808 ("the '808 Patent"), entitled "Method for Coding Motion in a Video Sequence," to inventor Jani Lainema. Nokia owns all rights to the '808 Patent necessary to bring this action. A true and correct copy of the '808 Patent is attached hereto as Exhibit 3 and incorporated herein by reference.

42. On September 27, 2005, the U.S. Patent and Trademark Office duly and legally issued U.S. Patent No. 6,950,469 ("the '469 Patent"), entitled "Method for Sub-Pixel Value Interpolation," to Marta Karczewicz and Antti Olli Hallapuro. Nokia owns all rights to the '469 Patent necessary to bring this action. A true and correct copy of the '469 Patent is attached hereto as Exhibit 4 and incorporated herein by reference.

43. On October 11, 2011, the U.S. Patent and Trademark Office duly and legally issued U.S. Patent No. 8,036,273 ("the '273 Patent"), entitled "Method for Sub-Pixel Value Interpolation," to Marta Karczewicz and Antti Olli Hallapuro. Nokia owns all rights to the '273 Patent necessary to bring this action. A true and correct copy of the '273 Patent is attached hereto as Exhibit 5 and incorporated herein by reference.

44. On March 27, 2012, the U.S. Patent and Trademark Office duly and legally issued U.S. Patent No. 8,144,764 ("the '764 Patent"), entitled "Video Coding," to Miska Hannuksela. Nokia owns all rights to the '764 Patent necessary to bring this action. A true and correct copy of the '764 Patent is attached hereto as Exhibit 6 and incorporated herein by reference.

45. On November 22, 2005, the U.S. Patent and Trademark Office duly and legally issued U.S. Patent No. 6,968,005 ("the '005 Patent"), entitled "Video coding," to Miska Hannuksela. Nokia owns all rights to the '005 Patent necessary to bring this action. A true and correct copy of the '005 Patent is attached hereto as Exhibit 7 and incorporated herein by reference.

46. On October 11, 2001, the U.S. Patent and Trademark Office duly and legally issued U.S. Patent No. 6,711,211 ("the '211 Patent"), entitled "Method for Encoding and Decoding Video Information, a Motion Compensated Video Encoder and a Corresponding Decoder," to Jani Lainema. Nokia owns all rights to the '211 Patent necessary to bring this action. A true and correct copy of the '211 Patent is attached hereto as Exhibit 8 and incorporated herein by reference.

47. On February 15, 2005, the U.S. Patent and Trademark Office duly and legally issued U.S. Patent No. 6,856,701 ("the '701 Patent"), entitled "Method and System for Context-Based Adaptive Binary Arithmetic Coding," to Marta Karczewicz and Ragip Kurceren. Nokia

owns all rights to the ’701 Patent necessary to bring this action. A true and correct copy of the ’701 Patent is attached hereto as Exhibit 9 and incorporated herein by reference.

48. On January 20, 2004, the U.S. Patent and Trademark Office duly and legally issued U.S. Patent No. 6,680,974 (“the ’974 Patent”), entitled “Methods and apparatus for context selection of block transform coefficients,” to Alireza Farid Faryar, Moushumi Sen, and Kyeong Ho Yang. Nokia owns all rights to the ’974 Patent necessary to bring this action. A true and correct copy of the ’974 Patent is attached hereto as Exhibit 10 and incorporated herein by reference.

49. The ’808, ’469, ’273, ’764, ’005, ’211, ’701, and ’974 Patents are collectively referred to as the “Nokia Patents-in-Suit.”

50. Nokia exclusively owns all rights, title, and interest in the Nokia Patents-in-Suit necessary to bring this action, including the right to recover past and future damages. Nokia has owned all rights to the Nokia Patents-in-Suit necessary to bring this action throughout the period of Apple’s infringement and still owns those rights to the Nokia Patents-in-Suit. Apple is not currently licensed to practice the Nokia Patents-in-Suit.

51. The Nokia Patents-in-Suit are valid and enforceable.

52. Apple has imported into the United States, manufactured, used, marketed, offered for sale, and/or sold in the United States, smartphones, tablets, and other mobile communication devices, computers, and digital media players that infringe the Nokia Patents-in-Suit.

53. Apple’s accused devices (“the Apple Accused Products”) which infringe one or more claims of the Nokia Patents-in-Suit include Apple products with H.264 video capabilities. The Accused Products include but are not limited to Apple’s iPhone, iPad, Apple TV, Mac

computer, and Apple Watch products and any other products capable of implementing the H.264 standard.

54. Apple has been placed on actual notice of infringement by Nokia prior to the filing of this Complaint as to certain of the Nokia Patents-in-Suit. At a minimum, in accordance with 35 U.S.C. § 287, Apple has had actual notice and knowledge of all the Nokia Patents-in-Suit at least as early as the filing of this Original Complaint and/or the date this Original Complaint was served upon Apple. Despite such notice, Apple continues to make, use, import into, market, offer for sale, and/or sell in the United States products that infringe the Nokia Patents-in-Suit.

GENERAL ALLEGATIONS

55. Apple has directly and indirectly infringed and continues to directly and indirectly infringe each of the Nokia Patents-in-Suit by engaging in acts constituting infringement under 35 U.S.C. § 271(a), (b), and/or (c), including but not necessarily limited to one or more of making, using, selling and offering to sell, in this District and elsewhere in the United States, and importing into the United States, the Apple Accused Products.

56. Apple's acts of infringement have caused damage to Nokia. Nokia is entitled to recover from Apple the damages sustained by Nokia as a result of Apple's wrongful acts in an amount subject to proof at trial.

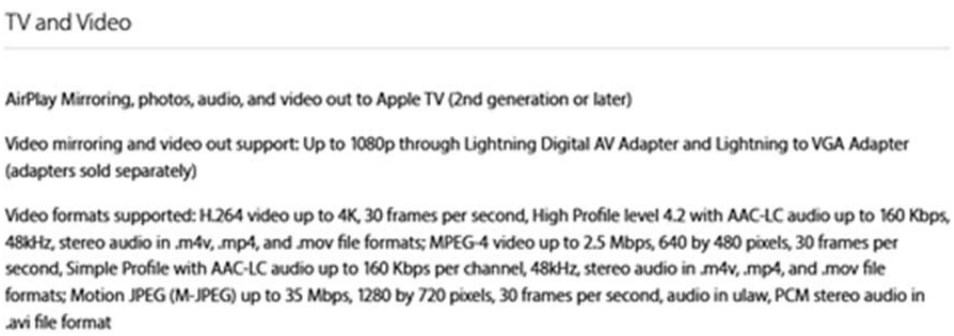
57. Apple's infringement of the Nokia Patents-in-Suit has been and continues to be willful. Apple has committed and continues to commit acts of infringement despite a high likelihood that its actions constitute infringement, and Apple knew or should have known that its actions constituted an unjustifiably high risk of infringement.

58. In the interest of providing detailed averments of infringement, Nokia has identified below at least one claim per patent to demonstrate infringement. However, the selection of claims should not be considered limiting, and additional claims of the Nokia Patents-in-Suit that are infringed by Apple will be disclosed in compliance with the Court's rules related to infringement contentions.

APPLE'S INFRINGING PRODUCTS

A. Apple Makes, Imports, Uses, Sells, and/or Offers for Sale Products and Services that Infringe the '808 Patent.

59. Each of the Accused Products is compliant with the H.264 Standard. For example, as shown below, Apple advertises that its iPhone 7 supports the H.264 video format.



TV and Video

AirPlay Mirroring, photos, audio, and video out to Apple TV (2nd generation or later)

Video mirroring and video out support: Up to 1080p through Lightning Digital AV Adapter and Lightning to VGA Adapter (adapters sold separately)

Video formats supported: H.264 video up to 4K, 30 frames per second, High Profile level 4.2 with AAC-LC audio up to 160 Kbps, 48kHz, stereo audio in .m4v, .mp4, and .mov file formats; MPEG-4 video up to 2.5 Mbps, 640 by 480 pixels, 30 frames per second, Simple Profile with AAC-LC audio up to 160 Kbps per channel, 48kHz, stereo audio in .m4v, .mp4, and .mov file formats; Motion JPEG (M-JPEG) up to 35 Mbps, 1280 by 720 pixels, 30 frames per second, audio in ulaw, PCM stereo audio in .avi file format

Source: http://www.apple.com/iphone-7/specs/.

60. Thus, for example and as shown below, the Accused Products infringe claim 16 of the '808 patent by virtue of their compatibility with and practice of the H.264 Standard.[2] For example, the Accused Products comprise a video decoder for decoding an encoded video sequence, the decoder comprising a demultiplexer for receiving an indication of a skip coding mode assigned to a first segment. As shown below, this functionality is described in the H.264 Standard, including but not limited to §§ 3, 7.3.4, 7.4.4, and 7.4.5.

[2] The infringement evidence cited herein includes exemplary citations to H.264 (03/2005). The cited functionality for each claim of infringement has been included in versions of the H.264 standard since March 2005 and remains in current versions of the H.264 standard.

3 Definitions

For the purposes of this Recommendation | International Standard, the following definitions apply.

. . .

3.75 macroblock: A 16x16 block of luma samples and two corresponding blocks of chroma samples . . .

. . .

3.135 skipped macroblock: A macroblock for which no data is coded other than an indication that the macroblock is to be decoded as "skipped". This indication may be common to several macroblocks.

. . .

Source: H.264 Standard at § 3.

slice_data( ) {	C	Descriptor
if(entropy_coding_mode_flag)		
while(!byte_aligned( ))		
cabac_alignment_one_bit	2	f(1)
CurrMbAddr = first_mb_in_slice * (1 + MbaffFrameFlag)		
moreDataFlag = 1		
prevMbSkipped = 0		
do {		
if(slice_type != I && slice_type != SI)		
if(!entropy_coding_mode_flag) {		
mb_skip_run	2	ue(v)
prevMbSkipped = (mb_skip_run > 0)		
for(i=0; i<mb_skip_run; i++)		
CurrMbAddr = NextMbAddress(CurrMbAddr)		
moreDataFlag = more_rbsp_data( )		
} else {		
mb_skip_flag	2	ae(v)
moreDataFlag = !mb_skip_flag		
}		

Source: H.264 Standard at § 7.3.4.

7.4.4 Slice data semantics

. . .

mb_skip_run specifies the number of consecutive skipped macroblocks for which, when decoding a P or SP slice, mb_type shall be inferred to be P_Skip and the macroblock type is collectively referred to as a P macroblock type . . .

. . .

mb_skip_flag equal to 1 specifies that for the current macroblock, when decoding a P or SP slice, mb_type shall be inferred to be P_Skip and the macroblock type is collectively referred to as P macroblock type . . .

Source: H.264 Standard at § 7.4.4.

7.4.5 Macroblock layer semantics

mb_type specifies the macroblock type. The semantics of mb_type depend on the slice type.

. . .

Macroblock types that may be collectively referred to as P macroblock types are specified in Table 7-13.

. . .

Table 7-13 – Macroblock type values 0 to 4 for P and SP slices

mb_type	Name of mb_type	NumMbPart (mb_type)	MbPartPredMode (mb_type, 0)	MbPartPredMode (mb_type, 1)	MbPartWidth (mb_type)	MbPartHeight (mb_type)
0	P_L0_16x16	1	Pred_L0	na	16	16
1	P_L0_L0_16x8	2	Pred_L0	Pred_L0	16	8
2	P_L0_L0_8x16	2	Pred_L0	Pred_L0	8	16
3	P_8x8	4	na	na	8	8
4	P_8x8ref0	4	na	na	8	8
inferred	P_Skip	1	Pred_L0	na	16	16

The following semantics are assigned to the macroblock types in Table 7-13.

. . .

– P_Skip: no further data is present for the macroblock in the bitstream.

Source: H.264 Standard at § 7.4.5.

61. The Accused Products further comprise a video decoder for decoding an encoded video sequence, the decoder further comprising a motion compensated prediction block for assigning either a zero motion vector or a predicted non-zero motion vector for the skip coding mode for the first segment based at least in part on the motion information of a second segment neighboring the first segment. As shown below, this functionality is described in the H.264 Standard, including but not limited to §§ 6.4.8, 8.4.1, 8.4.1.1, and 8.4.1.3.

> **6.4.8 Derivation processes for neighbouring macroblocks, blocks, and partitions**
>
> . . .
>
> Figure 6-14 illustrates the relative location of the neighbouring macroblocks, blocks, or partitions A, B, C, and D to the current macroblock, partition, or block, when the current macroblock, partition, or block is in frame coding mode.

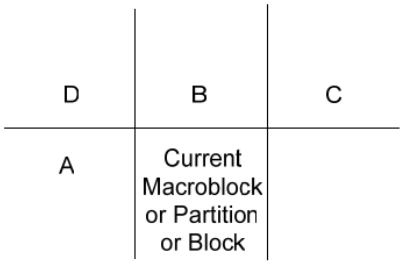


Figure 6-14 – Determination of the neighbouring macroblock, blocks, and partitions (informative)

Source: H.264 Standard at § 6.4.8.

> **8.4.1 Derivation process for motion vector components and reference indices**
>
> Inputs to this process are
>
> – a macroblock partition mbPartIdx,
>
> – a sub-macroblock partition subMbPartIdx.
>
> Outputs of this process are
>
> – luma motion vectors mvL0 and mvL1 as well as the chroma motion vectors mvCL0 and mvCL1
>
> – reference indices refIdxL0 and refIdxL1
>
> – prediction list utilization flags predFlagL0 and predFlagL1
>
> – a sub-partition macroblock motion vector count variable subMvCnt

For the derivation of the variables mvL0 and mvL1 as well as refIdxL0 and refIdxL1, the following applies.

– If mb_type is equal to P_Skip, the derivation process for luma motion vectors for skipped macroblocks in P and SP slices in subclause 8.4.1.1 is invoked with the output being the luma motion vectors mvL0 and reference indices refIdxL0, and predFlagL0 is set equal to 1. mvL1 and refIdxL1 are marked as not available and predFlagL1 is set equal to 0. The sub-partition motion vector count variable subMvCnt is set equal to 1.

– Otherwise, if mb_type is equal to B_Skip or B_Direct_16x16 or sub_mb_type[mbPartIdx] is equal to B_Direct_8x8, the derivation process for luma motion vectors for B_Skip, B_Direct_16x16, and B_Direct_8x8 in B slices in subclause 8.4.1.2 is invoked with mbPartIdx and subMbPartIdx as the input and the output being the luma motion vectors mvL0, mvL1, the reference indices refIdxL0, refIdxL1, the sub-partition motion vector count subMvCnt, and the prediction utilization flags predFlagL0 and predFlagL1.

– Otherwise, for X being replaced by either 0 or 1 in the variables predFlagLX, mvLX, refIdxLX, and in Pred_LX and in the syntax elements ref_idx_lX and mvd_lX, the following applies.

. . .

Source: H.264 Standard at § 8.4.1.

8.4.1.1 Derivation process for luma motion vectors for skipped macroblocks in P and SP slices

This process is invoked when mb_type is equal to P_Skip.

Outputs of this process are the motion vector mvL0 and the reference index refIdxL0.

The reference index refIdxL0 for a skipped macroblock is derived as follows.

refIdxL0 = 0.

For the derivation of the motion vector mvL0 of a P_Skip macroblock type, the following applies.

– The process specified in subclause 8.4.1.3.2 is invoked with mbPartIdx set equal to 0, subMbPartIdx set equal to 0, currSubMbType set equal to "na", and listSuffixFlag set equal to 0 as input and the output is assigned to mbAddrA, mbAddrB, mvL0A, mvL0B, refIdxL0A, and refIdxL0B.

– The variable mvL0 is specified as follows.

  – If any of the following conditions are true, both components of the motion vector mvL0 are set equal to 0.

    – mbAddrA is not available

    – mbAddrB is not available

- refIdxL0A is equal to 0 and both components of mvL0A are equal to 0
- refIdxL0B is equal to 0 and both components of mvL0B are equal to 0

– Otherwise, the derivation process for luma motion vector prediction as specified in subclause 8.4.1.3 is invoked with mbPartIdx = 0, subMbPartIdx = 0, refIdxL0, and currSubMbType = "na" as inputs and the output is assigned to mvL0.

NOTE – The output is directly assigned to mvL0, since the predictor is equal to the actual motion vector.

Source: H.264 Standard at § 8.4.1.1.

8.4.1.3 Derivation process for luma motion vector prediction

Inputs to this process are

– the macroblock partition index mbPartIdx,

– the sub-macroblock partition index subMbPartIdx,

– the reference index of the current partition refIdxLX (with X being 0 or 1),

– the variable currSubMbType.

Output of this process is the prediction mvpLX of the motion vector mvLX (with X being 0 or 1).

The derivation process for the neighbouring blocks for motion data in subclause 8.4.1.3.2 is invoked with mbPartIdx, subMbPartIdx, currSubMbType, and listSuffixFlag = X (with X being 0 or 1 for refIdxLX being refIdxL0 or refIdxL1, respectively) as the input and with mbAddrN\mbPartIdxN\subMbPartIdxN, reference indices refIdxLXN and the motion vectors mvLXN with N being replaced by A, B, or C as the output.

The derivation process for median luma motion vector prediction in subclause 8.4.1.3.1 is invoked with mbAddrN\mbPartIdxN\subMbPartIdxN, mvLXN, refIdxLXN with N being replaced by A, B, or C and refIdxLX as the input and mvpLX as the output, unless one of the following is true.

– MbPartWidth(mb_type) is equal to 16, MbPartHeight(mb_type) is equal to 8, mbPartIdx is equal to 0, and refIdxLXB is equal to refIdxLX,

mvpLX = mvLXB

– MbPartWidth(mb_type) is equal to 16, MbPartHeight(mb_type) is equal to 8, mbPartIdx is equal to 1, and refIdxLXA is equal to refIdxLX,

mvpLX = mvLXA

– MbPartWidth(mb_type) is equal to 8, MbPartHeight(mb_type) is equal to 16, mbPartIdx is equal to 0, and refIdxLXA is equal to refIdxLX,

mvpLX = mvLXA

– MbPartWidth(mb_type) is equal to 8, MbPartHeight(mb_type) is equal to 16, mbPartIdx is equal to 1, and refIdxLXC is equal to refIdxLX,

mvpLX = mvLXC

. . .

8.4.1.3.1 Derivation process for median luma motion vector prediction

Inputs to this process are

– the neighbouring partitions mbAddrN\mbPartIdxN\subMbPartIdxN (with N being replaced by A, B, or C),

– the motion vectors mvLXN (with N being replaced by A, B, or C) of the neighbouring partitions,

– the reference indices refIdxLXN (with N being replaced by A, B, or C) of the neighbouring partitions, and

– the reference index refIdxLX of the current partition.

Output of this process is the motion vector prediction mvpLX.

The variable mvpLX is derived as follows:

– When both partitions mbAddrB\mbPartIdxB\subMbPartIdxB and mbAddrC\mbPartIdxC\subMbPartIdxC are not available and mbAddrA\mbPartIdxA\subMbPartIdxA is available,

mvLXB = mvLXA

mvLXC = mvLXA

refIdxLXB = refIdxLXA

refIdxLXC = refIdxLXA

– Depending on reference indices refIdxLXA, refIdxLXB, or refIdxLXC, the following applies.

  – If one and only one of the reference indices refIdxLXA, refIdxLXB, or refIdxLXC is equal to the reference index refIdxLX of the current partition, the following applies. Let refIdxLXN be the reference index that is equal to refIdxLX, the motion vector mvLXN is assigned to the motion vector prediction mvpLX:

    mvpLX = mvLXN

  – Otherwise, each component of the motion vector prediction mvpLX is given by the median of the corresponding vector components of the motion vector mvLXA, mvLXB, and mvLXC:

    mvpLX[0] = Median(mvLXA[0], mvLXB[0], mvLXC[0])

    mvpLX[1] = Median(mvLXA[1], mvLXB[1], mvLXC[1])

8.4.1.3.2 Derivation process for motion data of neighbouring partitions

Inputs to this process are

– the macroblock partition index mbPartIdx,

– the sub-macroblock partition index subMbPartIdx,

– the current sub-macroblock type currSubMbType,

– the list suffix flag listSuffixFlag

Outputs of this process are (with N being replaced by A, B, or C)

– mbAddrN\mbPartIdxN\subMbPartIdxN specifying neighbouring partitions,

– the motion vectors mvLXN of the neighbouring partitions, and

– the reference indices refIdxLXN of the neighbouring partitions.

Variable names that include the string "LX" are interpreted with the X being equal to listSuffixFlag.

The partitions mbAddrN\mbPartIdxN\subMbPartIdxN with N being either A, B, or C are derived in the following

ordered steps.

1. Let mbAddrD\mbPartIdxD\subMbPartIdxD be variables specifying an additional neighbouring partition.
2. The process in subclause 6.4.8.5 is invoked with mbPartIdx, currSubMbType, and subMbPartIdx as input and the output is assigned to mbAddrN\mbPartIdxN\subMbPartIdxN with N being replaced by A, B, C, or D.
3. When the partition mbAddrC\mbPartIdxC\subMbPartIdxC is not available, the following applies

   mbAddrC = mbAddrD

   mbPartIdxC = mbPartIdxD

   subMbPartIdxC = subMbPartIdxD

The motion vectors mvLXN and reference indices refIdxLXN (with N being A, B, or C) are derived as follows.

– If the macroblock partition or sub-macroblock partition mbAddrN\mbPartIdxN\subMbPartIdxN is not available or mbAddrN is coded in Intra prediction mode or predFlagLX of mbAddrN\mbPartIdxN\subMbPartIdxN is equal to 0, both components of mvLXN are set equal to 0 and refIdxLXN is set equal to –1.

– Otherwise, the following applies.

  – The motion vector mvLXN and reference index refIdxLXN are set equal to MvLX[mbPartIdxN][subMbPartIdxN] and RefIdxLX[mbPartIdxN], respectively, which are the motion vector mvLX and reference index refIdxLX that have been assigned to the (sub-)macroblock partition mbAddrN\mbPartIdxN\subMbPartIdxN.

- The variables mvLXN[1] and refIdxLXN are further processed as follows.
  - If the current macroblock is a field macroblock and the macroblock mbAddrN is a frame macroblock

    mvLXN[1] = mvLXN[1] / 2

    refIdxLXN = refIdxLXN * 2
  - Otherwise, if the current macroblock is a frame macroblock and the macroblock mbAddrN is a field macroblock

    mvLXN[1] = mvLXN[1] * 2

    refIdxLXN = refIdxLXN / 2
  - Otherwise, the vertical motion vector component mvLXN[1] and the reference index refIdxLXN remain unchanged.

Source: H.264 Standard at § 8.4.1.3.

62. The Accused Products further comprise a video decoder for decoding an encoded video sequence, the decoder further comprising a motion compensated prediction block for forming a prediction for the first segment with respect to a reference frame based at least in part on the assigned motion vector for the skip coding mode, wherein the assigned motion vector is one of the zero motion vector and the predicted non-zero motion vector. As shown below, this functionality is described in the H.264 Standard, including but not limited to §§ 8.4 and 8.4.2.

8.4 Inter prediction process

This process is invoked when decoding P and B macroblock types.

Outputs of this process are Inter prediction samples for the current macroblock that are a 16x16 array predL of luma samples and when chroma_format_idc is not equal to 0 (monochrome) two 8x8 arrays predCr and predCb of chroma samples, one for each of the chroma components Cb and Cr.

The partitioning of a macroblock is specified by mb_type. Each macroblock partition is referred to by mbPartIdx. When the macroblock partitioning consists of partitions that are equal to sub-macroblocks, each sub-macroblock can be further partitioned into sub-macroblock partitions as specified by sub_mb_type. Each sub-macroblock partition is referred to by subMbPartIdx. When the macroblock partitioning does not consist of sub-macroblocks, subMbPartIdx is set equal to 0**. . .**

The Inter prediction process for a macroblock partition mbPartIdx and a sub-macroblock partition subMbPartIdx consists of the following ordered steps

. . .

1. Derivation process for motion vector components and reference indices as specified in subclause 8.4.1.

   Inputs to this process are

   – a macroblock partition mbPartIdx,

   – a sub-macroblock partition subMbPartIdx.

   Outputs of this process are

   – luma motion vectors mvL0 and mvL1 and when chroma_format_idc is not equal to 0 (monochrome) the chroma motion vectors mvCL0 and mvCL1

   – reference indices refIdxL0 and refIdxL1

   – prediction list utilization flags predFlagL0 and predFlagL1

   – the sub-macroblock partition motion vector count subMvCnt.

2. The variable MvCnt is incremented by subMvCnt.

3. Decoding process for Inter prediction samples as specified in subclause 8.4.2.

Inputs to this process are

– a macroblock partition mbPartIdx,

– a sub-macroblock partition subMbPartIdx.

– variables specifying partition width and height for luma and chroma (if available), partWidth, partHeight, partWidthC (if available), and partHeightC (if available)

– luma motion vectors mvL0 and mvL1 and when chroma_format_idc is not equal to 0 (monochrome) the chroma motion vectors mvCL0 and mvCL1

– reference indices refIdxL0 and refIdxL1

– prediction list utilization flags predFlagL0 and predFlagL1

Outputs of this process are

– inter prediction samples (pred); which are a (partWidth)x(partHeight) array predPartL of prediction luma samples and when chroma_format_idc is not equal to 0 (monochrome) two (partWidthC)x(partHeightC) arrays predPartCr, and predPartCb of prediction chroma samples, one for each of the chroma components Cb and Cr.

. . .

Source: H.264 Standard at § 8.4.

8.4.2 Decoding process for Inter prediction samples

Inputs to this process are

– a macroblock partition mbPartIdx,

– a sub-macroblock partition subMbPartIdx.

- variables specifying partition width and height for luma and chroma (if available), partWidth, partHeight, partWidthC (if available) and partHeightC (if available)
- luma motion vectors mvL0 and mvL1 and when chroma_format_idc is not equal to 0 (monochrome) chroma motion vectors mvCL0 and mvCL1
- reference indices refIdxL0 and refIdxL1

Outputs of this process are

- the Inter prediction samples predPart, which are a (partWidth)x(partHeight) array predPartL of prediction luma samples, and when chroma_format_idc is not equal to 0 (monochrome) two (partWidthC)x(partHeightC) arrays $predPart_{Cb}$, $predPart_{Cr}$ of prediction chroma samples, one for each of the chroma components Cb and Cr.

Source: H.264 Standard at § 8.4.2.

63. Thus, as described above the Accused Products, including the Apple iPhone 7, infringe one or more claims of the ’808 patent, including claim 16.

64. Apple provides instruction manuals that instruct the users of the Accused Products to use the Accused Products in a manner that infringes the ’808 patent. For example, Apple advertises the compatibility of the iPhone 7 with H.264 video file formats. *See* http://www.apple.com/iphone-7/specs/.

B. Apple Makes, Imports, Uses, Sells, and/or Offers for Sale Products and Services that Infringe the ’469 Patent.

65. Each of the Accused Products is compliant with the H.264 Standard. For example, as shown below, Apple advertises that its iPhone 7 supports the H.264 video format.

TV and Video

AirPlay Mirroring, photos, audio, and video out to Apple TV (2nd generation or later)

Video mirroring and video out support: Up to 1080p through Lightning Digital AV Adapter and Lightning to VGA Adapter (adapters sold separately)

Video formats supported: H.264 video up to 4K, 30 frames per second, High Profile level 4.2 with AAC-LC audio up to 160 Kbps, 48kHz, stereo audio in .m4v, .mp4, and .mov file formats; MPEG-4 video up to 2.5 Mbps, 640 by 480 pixels, 30 frames per second, Simple Profile with AAC-LC audio up to 160 Kbps per channel, 48kHz, stereo audio in .m4v, .mp4, and .mov file formats; Motion JPEG (M-JPEG) up to 35 Mbps, 1280 by 720 pixels, 30 frames per second, audio in ulaw, PCM stereo audio in .avi file format

Source: http://www.apple.com/iphone-7/specs/.

66. Thus, for example and as shown below, the Accused Products infringe claim 27 of the '469 patent by virtue of their compatibility with and practice of the H.264 Standard. For example, the Accused Products comprise a communications terminal including an H.264 decoder, which comprises a video coder for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations. As shown below, this functionality is described in the H.264 Standard, including but not limited to §§ 6.2 and 8.4.2.2.

6.2 Source, decoded, and output picture formats

This subclause specifies the relationship between source and decoded frames and fields that is given via the bitstream.

The video source that is represented by the bitstream is a sequence of either or both frames or fields (called collectively pictures) in decoding order.

The source and decoded pictures (frames or fields) are each comprised of one or more sample arrays:

- Luma (Y) only (monochrome), with or without an auxiliary array.
- Luma and two Chroma (YCbCr or YCgCo), with or without an auxiliary array.
- Green, Blue and Red (GBR, also known as RGB), with or without an auxiliary array.

Source: H.264 Standard at § 6.2.

The number of bits necessary for the representation of each of the samples in the luma and chroma arrays in a video sequence is in the range of 8 to 12, and the number of bits used in the luma array may differ from the number of bits used in the chroma arrays.

When the value of chroma_format_idc is equal to 1, the nominal vertical and horizontal relative locations of luma and chroma samples in frames are shown in Figure 6-1. Alternative chroma sample relative locations may be indicated in video usability information (see Annex E).

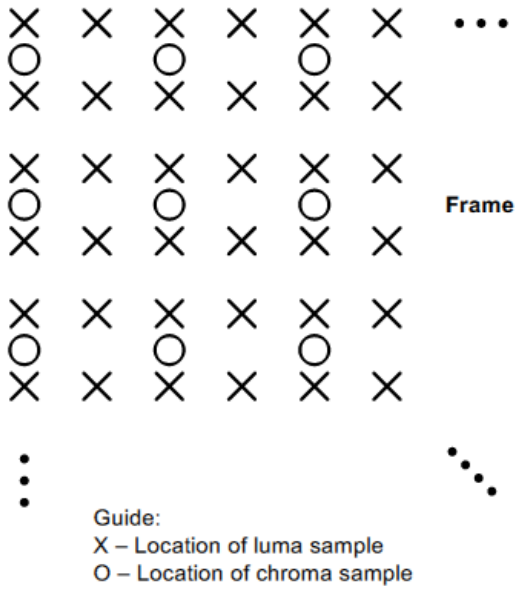


Figure 6-1 – Nominal vertical and horizontal locations of 4:2:0 luma and chroma samples in a frame

Source: H.264 Standard at § 6.2.

8.4.2.2.1 Luma sample interpolation process

Inputs to this process are

- a luma location in full-sample units ($xInt_L$, $yInt_L$),
- a luma location offset in fractional-sample units ($xFrac_L$, $yFrac_L$), and
- the luma sample array of the selected reference picture $refPicLX_L$

Output of this process is a predicted luma sample value $predPartLX_L[x_L, y_L]$.

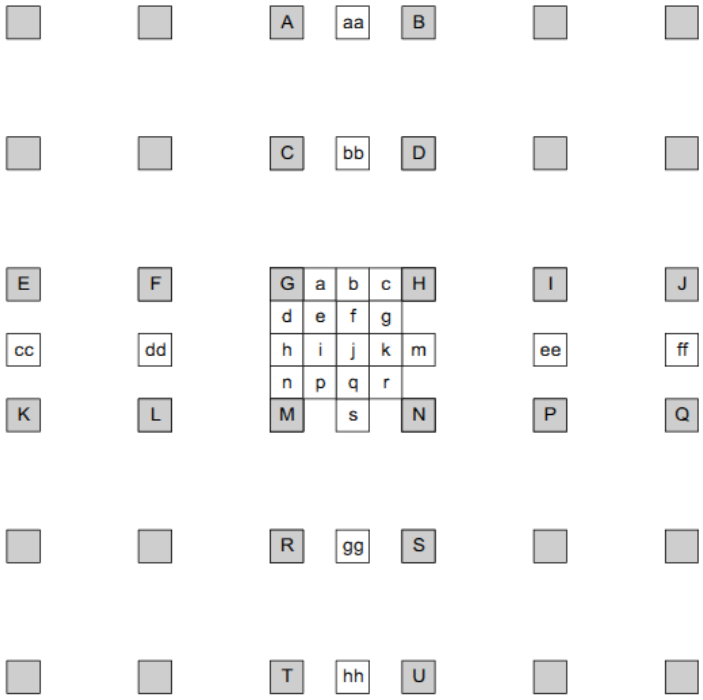


Figure 8-4 – Integer samples (shaded blocks with upper-case letters) and fractional sample positions (un-shaded blocks with lower-case letters) for quarter sample luma interpolation

Source: H.264 Standard at § 8.4.2.2.1.

67. The Accused Products further comprise the video coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the fractional horizontal and vertical locations being defined according to $½^x$, where x is a positive integer having a maximum value N. As shown below, this functionality is described in the H.264 Standard, including but not limited to § 8.4.2.2.

8.4.2.2 Fractional sample interpolation process

Inputs to this process are

- the current partition given by its partition index mbPartIdx and its sub-macroblock partition index subMbPartIdx,
- the width and height partWidth, partHeight of this partition in luma-sample units,
- a luma motion vector mvLX given in quarter-luma-sample units,
- a chroma motion vector mvCLX given in eighth-chroma-sample units, and
- the selected reference picture sample arrays refPicLX$_L$, refPicLX$_{Cb}$, and refPicLX$_{Cb}$

Outputs of this process are

- a (partWidth)x(partHeight) array predPartLX$_L$ of prediction luma sample values and
- when chroma_format_idc is not equal to 0 (monochrome) two (partWidthC)x(partHeightC) arrays predPartLX$_{Cb}$, and predPartLX$_{Cr}$ of prediction chroma sample values.

Source: H.264 Standard at § 8.4.2.2.

8.4.2.2.1 Luma sample interpolation process

Inputs to this process are

- a luma location in full-sample units (xInt$_L$, yInt$_L$),
- a luma location offset in fractional-sample units (xFrac$_L$, yFrac$_L$), and
- the luma sample array of the selected reference picture refPicLX$_L$

Output of this process is a predicted luma sample value predPartLX$_L$[x$_L$, y$_L$].

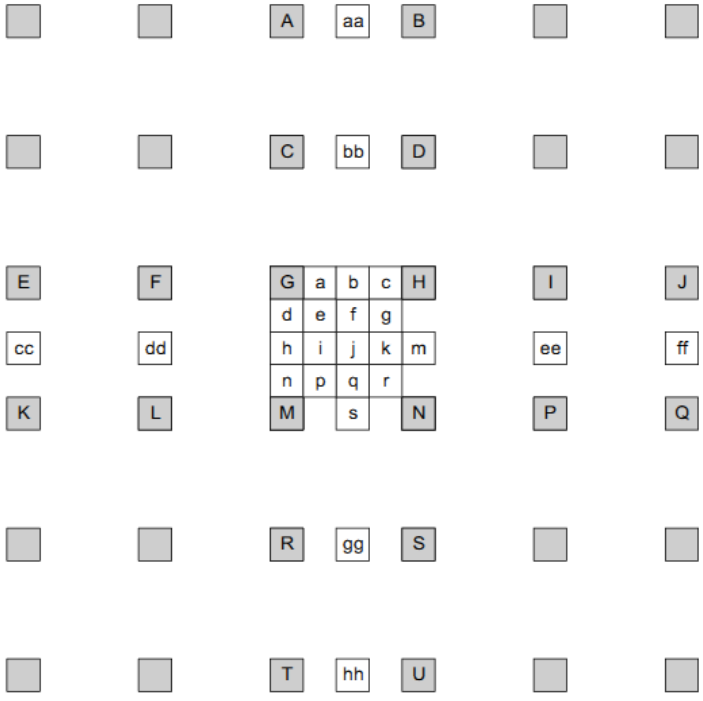


Figure 8-4 – Integer samples (shaded blocks with upper-case letters) and fractional sample positions (un-shaded blocks with lower-case letters) for quarter sample luma interpolation

Source: H.264 Standard at § 8.4.2.2.1.

68. The Accused Products further comprise the interpolator being adapted to (a) interpolate values for sub-pixels at $1/2^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $1/2^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations. As shown below, this functionality is described in the H.264 Standard, including but not limited to § 8.4.2.2.1.

– The samples at half sample positions labelled b are derived by first calculating intermediate values denoted as b_1 by applying the 6-tap filter to the nearest integer position samples in the horizontal direction. The samples at half sample positions labelled h are derived by first calculating intermediate values denoted as h_1 by applying the 6-tap filter to the nearest integer position samples in the vertical direction:

$$b_1 = (E - 5 * F + 20 * G + 20 * H - 5 * I + J) \quad (8\text{-}237)$$
$$h_1 = (A - 5 * C + 20 * G + 20 * M - 5 * R + T) \quad (8\text{-}238)$$

The final prediction values b and h are derived using:

$$b = Clip1_Y((b_1 + 16) >> 5) \quad (8\text{-}239)$$
$$h = Clip1_Y((h_1 + 16) >> 5) \quad (8\text{-}240)$$

Source: H.264 Standard at § 8.4.2.2.1.

69. The Accused Products further comprise the interpolator being adapted to (b) interpolate values for sub-pixels at $1/2^{N-1}$ unit horizontal and $1/2^{N-1}$ unit vertical locations directly using a choice of a first weighted sum of values for sub-pixels residing at $1/2^{N-1}$ unit horizontal and unit vertical locations and a second weighted sum of values for sub-pixels residing at unit horizontal and $1/2^{N-1}$ unit vertical locations, the first and second weighted sums of values being calculated according to step (a). As shown below, this functionality is described in the H.264 Standard, including but not limited to § 8.4.2.2.1.

– The samples at half sample position labelled as j are derived by first calculating intermediate value denoted as j_1 by applying the 6-tap filter to the intermediate values of the closest half sample positions in either the horizontal or vertical direction because these yield an equal result.

$$j_1 = cc - 5 * dd + 20 * h_1 + 20 * m_1 - 5 * ee + ff, \text{ or} \quad (8\text{-}241)$$
$$j_1 = aa - 5 * bb + 20 * b_1 + 20 * s_1 - 5 * gg + hh \quad (8\text{-}242)$$

where intermediate values denoted as aa, bb, gg, s_1 and hh are derived by applying the 6-tap filter horizontally in the same manner as the derivation of b_1 and intermediate values denoted as cc, dd, ee, m_1 and ff are derived by applying the 6-tap filter vertically in the same manner as the derivation of h_1. The final prediction value j are derived using:

$$j = Clip1_Y((j_1 + 512) >> 10) \quad (8\text{-}243)$$

Source: H.264 Standard at § 8.4.2.2.1.

70. The Accused Products further comprise the interpolator being adapted to (c) interpolate a value for a sub-pixel situated at a $½^N$ unit horizontal and $½^N$ unit vertical location by taking a weighted average of the value of a first sub-pixel or pixel situated at a $½^{N-m}$ unit horizontal and $½^{N-n}$ unit vertical location and the value of a second sub-pixel or pixel located at a $½^{N-p}$ unit horizontal and $½^{N-q}$ unit vertical location, variables m, n, p and q taking integer values in the range 1 to N such that the first and second sub-pixels or pixels are located diagonally with respect to the sub-pixel at $½^N$ unit horizontal and $½^N$ vertical location. As shown below, this functionality is described in the H.264 Standard, including but not limited to § 8.4.2.2.1.

– The samples at quarter sample positions labelled as e, g, p, and r are derived by averaging with upward rounding of the two nearest samples at half sample positions in the diagonal direction using

$$e = (b + h + 1) >> 1 \quad (8\text{-}254)$$
$$g = (b + m + 1) >> 1 \quad (8\text{-}255)$$
$$p = (h + s + 1) >> 1 \quad (8\text{-}256)$$
$$r = (m + s + 1) >> 1. \quad (8\text{-}257)$$

Source: H.264 Standard at § 8.4.2.2.1.

71. Thus, as described above, the Accused Products, including the Apple iPhone 7, infringe one or more claims of the ’469 patent, including claim 27.

72. Apple provides instruction manuals that instruct the users of the Accused Products to use the Accused Products in a manner that infringes the ’469 patent. For example,

Apple advertises the compatibility of the iPhone 7 with H.264 video file formats. *See* http://www.apple.com/iphone-7/specs/.

C. Apple Makes, Imports, Uses, Sells, and/or Offers for Sale Products and Services that Infringe the '273 Patent.

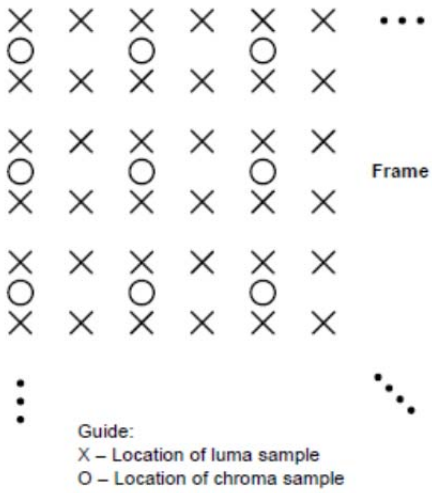
73. Each of the Accused Products is compliant with the H.264 technical standard. For example, as shown below, Apple advertises that its iPhone 7 supports the H.264 video format.

TV and Video	AirPlay Mirroring, photos, audio, and video out to Apple TV (2nd generation or later) Video mirroring and video out support: Up to 1080p through Lightning Digital AV Adapter and Lightning to VGA Adapter (adapters sold separately) Video formats supported: H.264 video up to 4K, 30 frames per second, High Profile level 4.2 with AAC-LC audio up to 160 Kbps, 48kHz, stereo audio or Dolby Audio up to 1008 Kbps, 48kHz, stereo or multichannel audio, in .m4v, .mp4, and .mov file formats; MPEG-4 video up to 2.5 Mbps, 640 by 480 pixels, 30 frames per second, Simple Profile with AAC-LC audio up to 160 Kbps per channel, 48kHz, stereo audio or Dolby Audio up to 1008 Kbps, 48kHz, stereo or multichannel audio, in .m4v, .mp4, and .mov file formats; Motion JPEG (M-JPEG) up to 35 Mbps, 1280 by 720 pixels, 30 frames per second, audio in ulaw, PCM stereo audio in .avi file format

Source: http://www.apple.com/iphone-7/specs/

74. Thus, for example and as shown below, the Accused Products infringe claim 33 of the '273 patent by virtue of their compatibility with and practice of the H.264 Standard. For example, the Accused Products comprise an interpolator for sub-pixel value interpolation, the interpolator being configured to determine values for sub-pixels situated within a rectangular bounded region defined by four corner pixels with no intermediate pixels between the corners, the pixels and sub-pixels being configured for display in rows and columns, pixel and sub-pixel locations in the rows and columns being representable mathematically within the rectangular bounded region using the co-ordinate notation $K/2^N$, $L/2^N$, K and L being positive integers having respective values between zero and 2^N, N being a positive integer greater than one and

representing a particular degree of sub-pixel value interpolation. This functionality is described in the H.264 Standard, including but not limited to §§ 6.2, 8.4.2.2 and 8.4.2.2.1.



Source: H.264 Standard at Figure 6-1.

8.4.2.2.1 Luma sample interpolation process

Inputs to this process are

- a luma location in full-sample units ($xInt_L$, $yInt_L$),
- a luma location offset in fractional-sample units ($xFrac_L$, $yFrac_L$), and
- the luma sample array of the selected reference picture $refPicLX_L$

Output of this process is a predicted luma sample value $predPartLX_L[x_L, y_L]$.

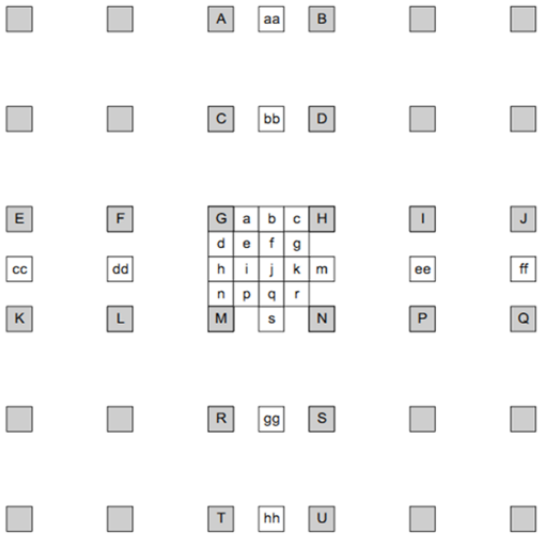


Figure 8-4 – Integer samples (shaded blocks with upper-case letters) and fractional sample positions (un-shaded blocks with lower-case letters) for quarter sample luma interpolation

Source: H.264 Standard at § 8.4.2.2.1

75. The Accused Products also interpolate a sub-pixel value for a sub-pixel having co-ordinates with odd values of K and L, according to a predetermined choice of either a weighted average of the value of a nearest-neighboring pixel and the value of the sub-pixel situated at co-ordinates ½, ½, or a weighted average of the values of a pair of diagonally-opposed sub-pixels having co-ordinates with even values of K and L, including zero, situated within a quadrant of the rectangular bounded region, the quadrant being defined by the sub-pixel having co-ordinates ½, ½ and the nearest neighboring pixel. This functionality is described in the H.264 Standard, including but not limited to § 8.4.2.2.1.

– The samples at quarter sample positions labelled as a, c, d, n, f, i, k, and q are derived by averaging with upward rounding of the two nearest samples at integer and half sample positions using:

$$a = (G + b + 1) >> 1 \quad (8\text{-}246)$$
$$c = (H + b + 1) >> 1 \quad (8\text{-}247)$$
$$d = (G + h + 1) >> 1 \quad (8\text{-}248)$$
$$n = (M + h + 1) >> 1 \quad (8\text{-}249)$$
$$f = (b + j + 1) >> 1 \quad (8\text{-}250)$$
$$i = (h + j + 1) >> 1 \quad (8\text{-}251)$$
$$k = (j + m + 1) >> 1 \quad (8\text{-}252)$$
$$q = (j + s + 1) >> 1. \quad (8\text{-}253)$$

– The samples at quarter sample positions labelled as e, g, p, and r are derived by averaging with upward rounding of the two nearest samples at half sample positions in the diagonal direction using

$$e = (b + h + 1) >> 1 \quad (8\text{-}254)$$
$$g = (b + m + 1) >> 1 \quad (8\text{-}255)$$
$$p = (h + s + 1) >> 1 \quad (8\text{-}256)$$
$$r = (m + s + 1) >> 1. \quad (8\text{-}257)$$

Source: H.264 Standard at § 8.4.2.2.1.

76. The Accused Products also interpolate sub-pixel values for sub-pixels having co-ordinates with K equal to an even value and L equal to zero and sub-pixels having co-ordinates with K equal to zero and L equal to an even value, used in the interpolation of the sub-pixels having co-ordinates with odd values of K and L, using weighted sums of the values of pixels located in rows and columns respectively. This functionality is described in the H.264 Standard, including but not limited to § 8.4.2.2.1.

– The samples at half sample positions labelled b are derived by first calculating intermediate values denoted as b_1 by applying the 6-tap filter to the nearest integer position samples in the horizontal direction. The samples at half sample positions labelled h are derived by first calculating intermediate values denoted as h_1 by applying the 6-tap filter to the nearest integer position samples in the vertical direction:

$$b_1 = (E - 5 * F + 20 * G + 20 * H - 5 * I + J) \quad (8\text{-}237)$$
$$h_1 = (A - 5 * C + 20 * G + 20 * M - 5 * R + T) \quad (8\text{-}238)$$

The final prediction values b and h are derived using:

$$b = Clip1_Y((b_1 + 16) >> 5) \quad (8\text{-}239)$$
$$h = Clip1_Y((h_1 + 16) >> 5) \quad (8\text{-}240)$$

Source: H.264 Standard at § 8.4.2.2.1.

77. The Accused Products also interpolate sub-pixel values for sub-pixels having co-ordinates with even values of K and L, used in the interpolation of sub-pixel values for the sub-pixels having co-ordinates with odd values of K and L, using a predetermined choice of either a weighted sum of the values of sub-pixels having co-ordinates with K equal to an even value and L equal to zero and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions, or a weighted sum of the values of sub-pixels having co-ordinates with K equal to zero and L equal to an even value and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions. This functionality is described in the H.264 Standard, including but not limited to § 8.4.2.2.1.

– The samples at half sample position labelled as j are derived by first calculating intermediate value denoted as j_1 by applying the 6-tap filter to the intermediate values of the closest half sample positions in either the horizontal or vertical direction because these yield an equal result.

$$j_1 = cc - 5 * dd + 20 * h_1 + 20 * m_1 - 5 * ee + ff, \text{ or} \quad (8\text{-}241)$$
$$j_1 = aa - 5 * bb + 20 * b_1 + 20 * s_1 - 5 * gg + hh \quad (8\text{-}242)$$

where intermediate values denoted as aa, bb, gg, s_1 and hh are derived by applying the 6-tap filter horizontally in the same manner as the derivation of b_1 and intermediate values denoted as cc, dd, ee, m_1 and ff are derived by applying the 6-tap filter vertically in the same manner as the derivation of h_1. The final prediction value j are derived using:

$$j = Clip1_Y((j_1 + 512) >> 10) \quad (8\text{-}243)$$

Source: H.264 Standard at § 8.4.2.2.1.

78. Thus, as described above the Accused Products, including the Apple iPhone 7, infringe one or more claims of the '273 patent, including claim 33.

79. Apple provides instruction manuals that instruct the users of the Accused Products to use the Accused Products in a manner that infringes the '273 patent. For example, Apple advertises the compatibility of the iPhone 7 with H.264 video file formats. *See* http://www.apple.com/iphone-7/specs/

D. Apple Makes, Imports, Uses, Sells, and/or Offers for Sale Products and Services that Infringe the '764 Patent.

80. Each of the Accused Products is compliant with the H.264 Standard. For example, as shown below, Apple advertises that its iPhone 7 supports the H.264 video format.

TV and Video

AirPlay Mirroring, photos, audio, and video out to Apple TV (2nd generation or later)

Video mirroring and video out support: Up to 1080p through Lightning Digital AV Adapter and Lightning to VGA Adapter (adapters sold separately)

Video formats supported: H.264 video up to 4K, 30 frames per second, High Profile level 4.2 with AAC-LC audio up to 160 Kbps, 48kHz, stereo audio in .m4v, .mp4, and .mov file formats; MPEG-4 video up to 2.5 Mbps, 640 by 480 pixels, 30 frames per second, Simple Profile with AAC-LC audio up to 160 Kbps per channel, 48kHz, stereo audio in .m4v, .mp4, and .mov file formats; Motion JPEG (M-JPEG) up to 35 Mbps, 1280 by 720 pixels, 30 frames per second, audio in ulaw, PCM stereo audio in .avi file format

Source: http://www.apple.com/iphone-7/specs/.

81. Thus, for example and as shown below, the Accused Products infringe claim 46 of the '764 patent by virtue of their compatibility with and practice of the H.264 Standard. For example, the Accused Products comprise an apparatus for decoding an encoded video signal representing a sequence of pictures to form a decoded video signal. As shown below, this functionality is described in the H.264 Standard, including but not limited to § 3.

3.1 **access unit**: A set of *NAL units* always containing exactly one *primary coded picture*. In addition to the *primary coded picture*, an access unit may also contain one or more *redundant coded pictures* or other *NAL units* not containing *slices* or *slice data partitions* of a *coded picture*. The decoding of an access unit always results in a *decoded picture*.

3.14 **bitstream**: A sequence of bits that forms the representation of *coded pictures* and associated data forming one or more *coded video sequences*. Bitstream is a collective term used to refer either to a *NAL unit stream* or a *byte stream*.

3.27 **coded picture**: A *coded representation* of a *picture*. A coded picture may be either a *coded field* or a *coded frame*. Coded picture is a collective term referring to a *primary coded picture* or a *redundant coded picture*, but not to both together.

3.30 **coded video sequence**: A sequence of *access units* that consists, in decoding order, of an *IDR access unit* followed by zero or more non-IDR *access units* including all subsequent *access units* up to but not including any subsequent *IDR access unit*.

3.37 **decoded picture**: A *decoded picture* is derived by decoding a *coded picture*. A *decoded picture* is either a decoded *frame*, or a decoded *field*. A decoded *field* is either a decoded *top field* or a decoded *bottom field*.

3.38 **decoded picture buffer (DPB)**: A buffer holding *decoded pictures* for reference, output reordering, or output delay specified for the *hypothetical reference decoder* in Annex C.

3.39 **decoder**: An embodiment of a *decoding process*.

3.40 **decoding order**: The order in which *syntax elements* are processed by the *decoding process*.

3.41 **decoding process**: The process specified in this Recommendation | International Standard that reads a *bitstream* and derives *decoded pictures* from it.

3.47 **encoding process**: A process, not specified in this Recommendation | International Standard, that produces a *bitstream* conforming to this Recommendation | International Standard.

3.48 **field**: An assembly of alternate rows of a *frame*. A *frame* is composed of two *fields*, a *top field* and a *bottom field*.

3.61 **instantaneous decoding refresh (IDR) access unit**: An *access unit* in which the *primary coded picture* is an *IDR picture*.

3.62 **instantaneous decoding refresh (IDR) picture**: A *coded picture* in which all *slices* are *I* or *SI slices* that causes the *decoding process* to mark all *reference pictures* as "unused for reference" immediately after decoding the IDR picture. After the decoding of an IDR picture all following *coded pictures* in *decoding order* can be decoded without *inter prediction* from any *picture* decoded prior to the IDR picture. The first *picture* of each *coded video sequence* is an IDR picture.

3.87 **NAL unit**: A syntax structure containing an indication of the type of data to follow and *bytes* containing that data in the form of an *RBSP* interspersed as necessary with *emulation prevention bytes*.

3.94 **non-reference picture**: A *picture* coded with nal_ref_idc equal to 0. A *non-reference picture* is not used for *inter prediction* of any other *pictures*.

3.102 **picture**: A collective term for a *field* or a *frame*.

3.109 **primary coded picture**: The coded representation of a *picture* to be used by the *decoding process* for a bitstream conforming to this Recommendation | International Standard. The primary coded picture contains all *macroblocks* of the *picture*. The only *pictures* that have a normative effect on the *decoding process* are primary coded pictures. See also *redundant coded picture*.

3.121 **reference picture**: A *picture* with nal_ref_idc not equal to 0. A *reference picture* contains samples that may be used for *inter prediction* in the *decoding process* of subsequent *pictures* in *decoding order*.

3.136 **slice**: An integer number of *macroblocks* or *macroblock pairs* ordered consecutively in the *raster scan* within a particular *slice group*. For the *primary coded picture*, the division of each *slice group* into slices is a *partitioning*. Although a slice contains *macroblocks* or *macroblock pairs* that are consecutive in the *raster scan* within a *slice group*, these *macroblocks* or *macroblock pairs* are not necessarily consecutive in the *raster scan* within the *picture*. The addresses of the *macroblocks* are derived from the address of the first *macroblock* in a slice (as represented in the *slice header*) and the *macroblock to slice group map*.

3.138 **slice group**: A subset of the *macroblocks* or *macroblock pairs* of a *picture*. The division of the *picture* into slice groups is a *partitioning* of the *picture*. The partitioning is specified by the *macroblock to slice group map*.

3.140 **slice header**: A part of a coded *slice* containing the data elements pertaining to the first or all *macroblocks* represented in the *slice*.

Source: H.264 Standard at § 3.

82. The Accused Products further comprise wherein the apparatus is configured to examine decoded reference pictures to identify a difference in respective sequence indicator values assigned to consecutively encoded reference pictures. As shown below, this functionality is described in the H.264 Standard, including but not limited to §§ 8, 8.1, 7.3.1, 7.4.1, 7.3.3, and 7.4.3.

8 Decoding process

Outputs of this process are decoded samples of the current picture (sometimes referred to by the variable CurrPic).

This clause describes the decoding process, given syntax elements and upper-case variables from clause 7.

The decoding process is specified such that all decoders shall produce numerically identical results. Any decoding process that produces identical results to the process described here conforms to the decoding process requirements of this Recommendation | International Standard.

Each picture referred to in this clause is a primary picture. Each slice referred to in this clause is a slice of a primary picture. Each slice data partition referred to in this clause is a slice data partition of a primary picture.

An overview of the decoding process is given as follows.

– The decoding of NAL units is specified in subclause 8.1.

– The processes in subclause 8.2 specify decoding processes using syntax elements in the slice layer and above.

Source: H.264 Standard at § 8.

8.1 NAL unit decoding process

Inputs to this process are NAL units.

Outputs of this process are the RBSP syntax structures encapsulated within the NAL units.

The decoding process for each NAL unit extracts the RBSP syntax structure from the NAL unit and then operates the decoding processes specified for the RBSP syntax structure in the NAL unit as follows.

Source: H.264 Standard at § 8.1.

7.3.1 NAL unit syntax

nal_unit(NumBytesInNALunit) {	C	Descriptor
forbidden_zero_bit	All	f(1)
nal_ref_idc	All	u(2)
nal_unit_type	All	u(5)
NumBytesInRBSP = 0		
for(i = 1; i < NumBytesInNALunit; i++) {		
if(i + 2 < NumBytesInNALunit && next_bits(24) == 0x000003) {		
rbsp_byte[NumBytesInRBSP++ **]**	All	b(8)
rbsp_byte[NumBytesInRBSP++ **]**	All	b(8)
i += 2		
emulation_prevention_three_byte /* equal to 0x03 */	All	f(8)
} else		
rbsp_byte[NumBytesInRBSP++ **]**	All	b(8)
}		
}		

Source: H.264 Standard at § 7.3.1.

nal_ref_idc not equal to 0 specifies that the content of the NAL unit contains a sequence parameter set or a picture parameter set or a slice of a reference picture or a slice data partition of a reference picture.

nal_ref_idc equal to 0 for a NAL unit containing a slice or slice data partition indicates that the slice or slice data partition is part of a non-reference picture.

Source: H.264 Standard at § 7.4.1.

7.3.3 Slice header syntax

slice_header( ) {	C	Descriptor
first_mb_in_slice	2	ue(v)
slice_type	2	ue(v)
pic_parameter_set_id	2	ue(v)
frame_num	2	u(v)

Source: H.264 Standard at § 7.3.3.

frame_num is used as an identifier for pictures and shall be represented by log2_max_frame_num_minus4 + 4 bits in the bitstream. frame_num is constrained as follows:

The variable PrevRefFrameNum is derived as follows.

- If the current picture is an IDR picture, PrevRefFrameNum is set equal to 0.
- Otherwise (the current picture is not an IDR picture), PrevRefFrameNum is set as follows.
  - If the decoding process for gaps in frame_num specified in subclause 8.2.5.2 was invoked by the decoding process for an access unit that contained a non-reference picture that followed the previous access unit in decoding order that contained a reference picture, PrevRefFrameNum is set equal to the value of frame_num for the last of the "non-existing" reference frames inferred by the decoding process for gaps in frame_num specified in subclause 8.2.5.2.
  - Otherwise, PrevRefFrameNum is set equal to the value of frame_num for the previous access unit in decoding order that contained a reference picture.

The value of frame_num is constrained as follows.

- If the current picture is an IDR picture, frame_num shall be equal to 0.
- Otherwise (the current picture is not an IDR picture), referring to the primary coded picture in the previous access unit in decoding order that contains a reference picture as the preceding reference picture, the value of frame_num for the current picture shall not be equal to PrevRefFrameNum unless all of the following three conditions are true.
  - the current picture and the preceding reference picture belong to consecutive access units in decoding order
  - the current picture and the preceding reference picture are reference fields having opposite parity
  - one or more of the following conditions is true
    - the preceding reference picture is an IDR picture
    - the preceding reference picture includes a memory_management_control_operation syntax element equal to 5

      NOTE 1 – When the preceding reference picture includes a memory_management_control_operation syntax element equal to 5, PrevRefFrameNum is equal to 0.
    - there is a primary coded picture that precedes the preceding reference picture and the primary coded picture that precedes the preceding reference picture does not have frame_num equal to PrevRefFrameNum
    - there is a primary coded picture that precedes the preceding reference picture and the primary coded picture that precedes the preceding reference picture is not a reference picture

When the value of frame_num is not equal to PrevRefFrameNum, the following applies.

- There shall not be any previous field or frame in decoding order that is currently marked as "used for short-term reference" that has a value of frame_num equal to any value taken on by the variable UnusedShortTermFrameNum in the following:

```
UnusedShortTermFrameNum = ( PrevRefFrameNum + 1 ) % MaxFrameNum
while( UnusedShortTermFrameNum  !=  frame_num )                    (7-21)
    UnusedShortTermFrameNum = ( UnusedShortTermFrameNum + 1 ) % MaxFrameNum
```

- The value of frame_num is constrained as follows.
  - If gaps_in_frame_num_value_allowed_flag is equal to 0, the value of frame_num for the current picture shall be equal to (PrevRefFrameNum + 1) % MaxFrameNum.

Source: H.264 Standard at § 7.4.3.

83. The Accused Products further comprise wherein the apparatus is configured to compare the identified difference in sequence indicator values on the basis of an independent numbering scheme in which consecutive reference pictures in encoding order are assigned sequence indicator values that differ with respect to each other by a predetermined amount, independent of one or more of the number of non-reference pictures encoded between consecutive reference pictures, and the number of non-coded pictures between consecutive reference pictures. As shown below, this functionality is described in the H.264 Standard, including but not limited to §§ 7.4.3 and 7.4.2.1.

frame_num is used as an identifier for pictures and shall be represented by log2_max_frame_num_minus4 + 4 bits in the bitstream. frame_num is constrained as follows:

The variable PrevRefFrameNum is derived as follows.

- If the current picture is an IDR picture, PrevRefFrameNum is set equal to 0.
- Otherwise (the current picture is not an IDR picture), PrevRefFrameNum is set as follows.
  - If the decoding process for gaps in frame_num specified in subclause 8.2.5.2 was invoked by the decoding process for an access unit that contained a non-reference picture that followed the previous access unit in decoding order that contained a reference picture, PrevRefFrameNum is set equal to the value of frame_num for the last of the "non-existing" reference frames inferred by the decoding process for gaps in frame_num specified in subclause 8.2.5.2.
  - Otherwise, PrevRefFrameNum is set equal to the value of frame_num for the previous access unit in decoding order that contained a reference picture.

The value of frame_num is constrained as follows.

- If the current picture is an IDR picture, frame_num shall be equal to 0.
- Otherwise (the current picture is not an IDR picture), referring to the primary coded picture in the previous access unit in decoding order that contains a reference picture as the preceding reference picture, the value of frame_num for the current picture shall not be equal to PrevRefFrameNum unless all of the following three conditions are true.
  - the current picture and the preceding reference picture belong to consecutive access units in decoding order
  - the current picture and the preceding reference picture are reference fields having opposite parity
  - one or more of the following conditions is true
    - the preceding reference picture is an IDR picture
    - the preceding reference picture includes a memory_management_control_operation syntax element equal to 5

      NOTE 1 – When the preceding reference picture includes a memory_management_control_operation syntax element equal to 5, PrevRefFrameNum is equal to 0.
    - there is a primary coded picture that precedes the preceding reference picture and the primary coded picture that precedes the preceding reference picture does not have frame_num equal to PrevRefFrameNum
    - there is a primary coded picture that precedes the preceding reference picture and the primary coded picture that precedes the preceding reference picture is not a reference picture

When the value of frame_num is not equal to PrevRefFrameNum, the following applies.

- There shall not be any previous field or frame in decoding order that is currently marked as "used for short-term reference" that has a value of frame_num equal to any value taken on by the variable UnusedShortTermFrameNum in the following:

```
UnusedShortTermFrameNum = ( PrevRefFrameNum + 1 ) % MaxFrameNum
while( UnusedShortTermFrameNum  !=  frame_num )                    (7-21)
    UnusedShortTermFrameNum = ( UnusedShortTermFrameNum + 1 ) % MaxFrameNum
```

- The value of frame_num is constrained as follows.
  - If gaps_in_frame_num_value_allowed_flag is equal to 0, the value of frame_num for the current picture shall be equal to (PrevRefFrameNum + 1) % MaxFrameNum.

Source: H.264 Standard at § 7.4.3.

gaps_in_frame_num_value_allowed_flag specifies the allowed values of frame_num as specified in subclause 7.4.3 and the decoding process in case of an inferred gap between values of frame_num as specified in subclause 8.2.5.2.

Source: H.264 Standard at § 7.4.2.1.

84. The Accused Products further comprise wherein the apparatus is configured to detect corruption or loss of a reference picture if said identified difference in sequence indicator values is more than said predetermined amount. As shown below, this functionality is described in the H.264 Standard, including but not limited to § 8.2.5.2.

8.2.5.2 Decoding process for gaps in frame_num

This process is invoked when frame_num is not equal to PrevRefFrameNum and is not equal to (PrevRefFrameNum + 1) % MaxFrameNum.

NOTE 1 – Although this process is specified as a subclause within subclause 8.2.5 (which defines a process that is invoked only when nal_ref_idc is not equal to 0), this process may also be invoked when nal_ref_idc is equal to 0 (as specified in clause 8). The reasons for the location of this subclause within the structure of this Recommendation | International Standard are historical.

NOTE 2 – This process can only be invoked for a conforming bitstream when gaps_in_frame_num_value_allowed_flag is equal to 1. When gaps_in_frame_num_value_allowed_flag is equal to 0 and frame_num is not equal to PrevRefFrameNum and is not equal to (PrevRefFrameNum + 1) % MaxFrameNum, the decoding process should infer an unintentional loss of pictures.

Source: H.264 Standard at § 8.2.5.2.

85. Thus, as described above, the Accused Products, including the Apple iPhone 7, infringe one or more claims of the '764 patent, including claim 46.

86. Apple provides instruction manuals that instruct the users of the Accused Products to use the Accused Products in a manner that infringes the '764 patent. For example, Apple advertises the compatibility of the iPhone 7 with H.264 video file formats. *See* http://www.apple.com/iphone-7/specs/.

E. Apple Makes, Imports, Uses, Sells, and/or Offers for Sale Products and Services that Infringe the '005 Patent.

87. Each of the Accused Products is compliant with the H.264 technical standard. For example, as shown below, Apple advertises that its iPhone 7 supports the H.264 video format.

TV and Video	AirPlay Mirroring, photos, audio, and video out to Apple TV (2nd generation or later) Video mirroring and video out support: Up to 1080p through Lightning Digital AV Adapter and Lightning to VGA Adapter (adapters sold separately) Video formats supported: H.264 video up to 4K, 30 frames per second, High Profile level 4.2 with AAC-LC audio up to 160 Kbps, 48kHz, stereo audio or Dolby Audio up to 1008 Kbps, 48kHz, stereo or multichannel audio, in .m4v, .mp4, and .mov file formats; MPEG-4 video up to 2.5 Mbps, 640 by 480 pixels, 30 frames per second, Simple Profile with AAC-LC audio up to 160 Kbps per channel, 48kHz, stereo audio or Dolby Audio up to 1008 Kbps, 48kHz, stereo or multichannel audio, in .m4v, .mp4, and .mov file formats; Motion JPEG (M-JPEG) up to 35 Mbps, 1280 by 720 pixels, 30 frames per second, audio in ulaw, PCM stereo audio in .avi file format

Source: http://www.apple.com/iphone-7/specs/

88. Thus, for example and as shown below, the Accused Products infringe claim 9 of the ’005 patent by virtue of their compatibility with and practice of the H.264 Standard. For example, the Accused Products comprise a video decoder for decoding an encoded video signal representing a sequence of pictures to form a decoded video signal, the encoded video signal comprising temporally independent INTRA pictures and temporally predicted pictures, wherein the INTRA pictures and at least some of the temporally predicted pictures form reference pictures for the temporal prediction of other pictures. This functionality is described in the H.264 Standard, including but not limited to §§ 3 and 7.4, and Table 7-6.

3.1 **access unit**: A set of *NAL units* always containing exactly one *primary coded picture*. In addition to the *primary coded picture*, an access unit may also contain one or more *redundant coded pictures* or other *NAL units* not containing *slices* or *slice data partitions* of a *coded picture*. The decoding of an access unit always results in a *decoded picture*.

3.14 **bitstream**: A sequence of bits that forms the representation of *coded pictures* and associated data forming one or more *coded video sequences*. Bitstream is a collective term used to refer either to a *NAL unit stream* or a *byte stream*.

3.27 **coded picture**: A *coded representation* of a *picture*. A coded picture may be either a *coded field* or a *coded frame*. Coded picture is a collective term referring to a *primary coded picture* or a *redundant coded picture*, but not to both together.

3.30 **coded video sequence**: A sequence of *access units* that consists, in decoding order, of an *IDR access unit* followed by zero or more non-IDR *access units* including all subsequent *access units* up to but not including any subsequent *IDR access unit*.

3.37 **decoded picture**: A *decoded picture* is derived by decoding a *coded picture*. A *decoded picture* is either a decoded *frame*, or a decoded *field*. A decoded *field* is either a decoded *top field* or a decoded *bottom field*.

3.38 **decoded picture buffer (DPB)**: A buffer holding *decoded pictures* for reference, output reordering, or output delay specified for the *hypothetical reference decoder* in Annex C.

3.39 **decoder**: An embodiment of a *decoding process*.

3.40 **decoding order**: The order in which *syntax elements* are processed by the *decoding process*.

3.41 **decoding process**: The process specified in this Recommendation | International Standard that reads a *bitstream* and derives *decoded pictures* from it.

3.47 **encoding process**: A process, not specified in this Recommendation | International Standard, that produces a *bitstream* conforming to this Recommendation | International Standard.

3.48 **field**: An assembly of alternate rows of a *frame*. A *frame* is composed of two *fields*, a *top field* and a *bottom field*.

3.61 **instantaneous decoding refresh (IDR) access unit**: An *access unit* in which the *primary coded picture* is an *IDR picture*.

3.62 **instantaneous decoding refresh (IDR) picture**: A *coded picture* in which all *slices* are *I* or *SI slices* that causes the *decoding process* to mark all *reference pictures* as "unused for reference" immediately after decoding the IDR picture. After the decoding of an IDR picture all following *coded pictures* in *decoding order* can be decoded without *inter prediction* from any *picture* decoded prior to the IDR picture. The first *picture* of each *coded video sequence* is an IDR picture.

3.87 **NAL unit**: A syntax structure containing an indication of the type of data to follow and *bytes* containing that data in the form of an *RBSP* interspersed as necessary with *emulation prevention bytes*.

3.94 **non-reference picture**: A *picture* coded with nal_ref_idc equal to 0. A *non-reference picture* is not used for *inter prediction* of any other *pictures*.

3.102 **picture**: A collective term for a *field* or a *frame*.

3.109 **primary coded picture**: The coded representation of a *picture* to be used by the *decoding process* for a bitstream conforming to this Recommendation | International Standard. The primary coded picture contains all *macroblocks* of the *picture*. The only *pictures* that have a normative effect on the *decoding process* are primary coded pictures. See also *redundant coded picture*.

3.121 **reference picture**: A *picture* with nal_ref_idc not equal to 0. A *reference picture* contains samples that may be used for *inter prediction* in the *decoding process* of subsequent *pictures* in *decoding order*.

3.136 **slice**: An integer number of *macroblocks* or *macroblock pairs* ordered consecutively in the *raster scan* within a particular *slice group*. For the *primary coded picture*, the division of each *slice group* into slices is a *partitioning*. Although a slice contains *macroblocks* or *macroblock pairs* that are consecutive in the *raster scan* within a *slice group*, these *macroblocks* or *macroblock pairs* are not necessarily consecutive in the *raster scan* within the *picture*. The addresses of the *macroblocks* are derived from the address of the first *macroblock* in a slice (as represented in the *slice header*) and the *macroblock to slice group map*.

3.138 **slice group**: A subset of the *macroblocks* or *macroblock pairs* of a *picture*. The division of the *picture* into slice groups is a *partitioning* of the *picture*. The partitioning is specified by the *macroblock to slice group map*.

3.140 **slice header**: A part of a coded *slice* containing the data elements pertaining to the first or all *macroblocks* represented in the *slice*.

Source: H.264 Standard at § 3.

3.63 **inter coding**: Coding of a *block*, *macroblock*, *slice*, or *picture* that uses *inter prediction*.

3.64 **inter prediction**: A *prediction* derived from decoded samples of *reference pictures* other than the current *decoded picture*.

3.66 **intra coding**: Coding of a *block*, *macroblock*, *slice*, or *picture* that uses *intra prediction*.

3.67 **intra prediction**: A *prediction* derived from the decoded samples of the same decoded *slice*.

3.68 **intra slice**: See *I slice*.

3.59 **I slice**: A *slice* that is not an *SI slice* that is decoded using *prediction* only from decoded samples within the same *slice*.

3.98 **P slice**: A *slice* that may be decoded using *intra prediction* from decoded samples within the same *slice* or *inter prediction* from previously-decoded *reference pictures*, using at most one *motion vector* and *reference index* to *predict* the sample values of each *block*.

3.8 **B slice**: A *slice* that may be decoded using *intra prediction* from decoded samples within the same *slice* or *inter prediction* from previously-decoded *reference pictures*, using at most two *motion vectors* and *reference indices* to *predict* the sample values of each *block*.

Source: H.264 Standard at § 3.

nal_ref_idc not equal to 0 specifies that the content of the NAL unit contains a sequence parameter set or a picture parameter set or a slice of a reference picture or a slice data partition of a reference picture.

nal_ref_idc equal to 0 for a NAL unit containing a slice or slice data partition indicates that the slice or slice data partition is part of a non-reference picture.

nal_ref_idc shall not be equal to 0 for sequence parameter set or sequence parameter set extension or picture parameter set NAL units. When nal_ref_idc is equal to 0 for one slice or slice data partition NAL unit of a particular picture, it shall be equal to 0 for all slice and slice data partition NAL units of the picture.

nal_ref_idc shall not be equal to 0 for IDR NAL units, i.e., NAL units with nal_unit_type equal to 5.

nal_ref_idc shall be equal to 0 for all NAL units having nal_unit_type equal to 6, 9, 10, 11, or 12.

Source: H.264 Standard at §7.4.1

7.4.2.4 Access unit delimiter RBSP semantics

The access unit delimiter may be used to indicate the type of slices present in a primary coded picture and to simplify the detection of the boundary between access units. There is no normative decoding process associated with the access unit delimiter.

primary_pic_type indicates that the slice_type values for all slices of the primary coded picture are members of the set listed in Table 7-5 for the given value of primary_pic_type.

Source: H.264 Standard at §7.4.2.4

7.4.3 Slice header semantics

When present, the value of the slice header syntax elements pic_parameter_set_id, frame_num, field_pic_flag, bottom_field_flag, idr_pic_id, pic_order_cnt_lsb, delta_pic_order_cnt_bottom, delta_pic_order_cnt[0], delta_pic_order_cnt[1], sp_for_switch_flag, and slice_group_change_cycle shall be the same in all slice headers of a coded picture.

first_mb_in_slice specifies the address of the first macroblock in the slice. When arbitrary slice order is not allowed as specified in Annex A, the value of first_mb_in_slice shall not be less than the value of first_mb_in_slice for any other slice of the current picture that precedes the current slice in decoding order.

The first macroblock address of the slice is derived as follows.

- If MbaffFrameFlag is equal to 0, first_mb_in_slice is the macroblock address of the first macroblock in the slice, and first_mb_in_slice shall be in the range of 0 to PicSizeInMbs - 1, inclusive.
- Otherwise (MbaffFrameFlag is equal to 1), first_mb_in_slice * 2 is the macroblock address of the first macroblock in the slice, which is the top macroblock of the first macroblock pair in the slice, and first_mb_in_slice shall be in the range of 0 to PicSizeInMbs / 2 - 1, inclusive.

slice_type specifies the coding type of the slice according to Table 7-6.

Source: H.264 Standard at §7.4.3

slice_type	Name of slice_type
0	P (P slice)
1	B (B slice)
2	I (I slice)
3	SP (SP slice)
4	SI (SI slice)
5	P (P slice)
6	B (B slice)
7	I (I slice)
8	SP (SP slice)
9	SI (SI slice)

Source: H.264 Standard at Table 7-6.

89. The Accused Products further comprise a video decoder for decoding an encoded video signal, the encoded video signal further comprising a sequence indicator having an independent numbering scheme such that consecutive reference pictures in encoding order are assigned sequence indicator values that differ with respect to each other by a predetermined amount independent of the number of non-reference pictures encoded between successive reference pictures. This functionality is described in the H.264 Standard, including but not limited to § 7.4.

frame_num is used as an identifier for pictures and shall be represented by log2_max_frame_num_minus4 + 4 bits in the bitstream. frame_num is constrained as follows:

The variable PrevRefFrameNum is derived as follows.

- If the current picture is an IDR picture, PrevRefFrameNum is set equal to 0.
- Otherwise (the current picture is not an IDR picture), PrevRefFrameNum is set as follows.
  - If the decoding process for gaps in frame_num specified in subclause 8.2.5.2 was invoked by the decoding process for an access unit that contained a non-reference picture that followed the previous access unit in decoding order that contained a reference picture, PrevRefFrameNum is set equal to the value of frame_num for the last of the "non-existing" reference frames inferred by the decoding process for gaps in frame_num specified in subclause 8.2.5.2.
  - Otherwise, PrevRefFrameNum is set equal to the value of frame_num for the previous access unit in decoding order that contained a reference picture.

The value of frame_num is constrained as follows.

- If the current picture is an IDR picture, frame_num shall be equal to 0.
- Otherwise (the current picture is not an IDR picture), referring to the primary coded picture in the previous access unit in decoding order that contains a reference picture as the preceding reference picture, the value of frame_num for the current picture shall not be equal to PrevRefFrameNum unless all of the following three conditions are true.
  - the current picture and the preceding reference picture belong to consecutive access units in decoding order
  - the current picture and the preceding reference picture are reference fields having opposite parity
  - one or more of the following conditions is true
    - the preceding reference picture is an IDR picture
    - the preceding reference picture includes a memory_management_control_operation syntax element equal to 5

      NOTE 1 – When the preceding reference picture includes a memory_management_control_operation syntax element equal to 5, PrevRefFrameNum is equal to 0.
    - there is a primary coded picture that precedes the preceding reference picture and the primary coded picture that precedes the preceding reference picture does not have frame_num equal to PrevRefFrameNum
    - there is a primary coded picture that precedes the preceding reference picture and the primary coded picture that precedes the preceding reference picture is not a reference picture

When the value of frame_num is not equal to PrevRefFrameNum, the following applies.

- There shall not be any previous field or frame in decoding order that is currently marked as "used for short-term reference" that has a value of frame_num equal to any value taken on by the variable UnusedShortTermFrameNum in the following:

```
UnusedShortTermFrameNum = ( PrevRefFrameNum + 1 ) % MaxFrameNum
while( UnusedShortTermFrameNum != frame_num )                    (7-21)
    UnusedShortTermFrameNum = ( UnusedShortTermFrameNum + 1 ) % MaxFrameNum
```

- The value of frame_num is constrained as follows.
  - If gaps_in_frame_num_value_allowed_flag is equal to 0, the value of frame_num for the current picture shall be equal to (PrevRefFrameNum + 1) % MaxFrameNum.

Source: H.264 Standard at § 7.4.3

90. The Accused Products further comprise an input for receiving the encoded video signal and being arranged to decode received encoded pictures, to examine each decoded picture that forms a reference picture to identify the sequence indicator value assigned to the reference picture and to compare the sequence indicator values assigned to consecutively decoded reference pictures to detect loss of a reference picture. This functionality is described in the H.264 Standard, including but not limited to §§ 8 and 8.2.5.2.

– When the frame_num of the current picture is not equal to PrevRefFrameNum and is not equal to (PrevRefFrameNum + 1) % MaxFrameNum, the decoding process for gaps in frame_num is performed according to subclause 8.2.5.2 prior to the decoding of any slices of the current picture.

Source: H.264 Standard at § 8.

8.2.5.2 Decoding process for gaps in frame_num

This process is invoked when frame_num is not equal to PrevRefFrameNum and is not equal to (PrevRefFrameNum + 1) % MaxFrameNum.

NOTE 1 – Although this process is specified as a subclause within subclause 8.2.5 (which defines a process that is invoked only when nal_ref_idc is not equal to 0), this process may also be invoked when nal_ref_idc is equal to 0 (as specified in clause 8). The reasons for the location of this subclause within the structure of this Recommendation | International Standard are historical.

NOTE 2 – This process can only be invoked for a conforming bitstream when gaps_in_frame_num_value_allowed_flag is equal to 1. When gaps_in_frame_num_value_allowed_flag is equal to 0 and frame_num is not equal to PrevRefFrameNum and is not equal to (PrevRefFrameNum + 1) % MaxFrameNum, the decoding process should infer an unintentional loss of pictures.

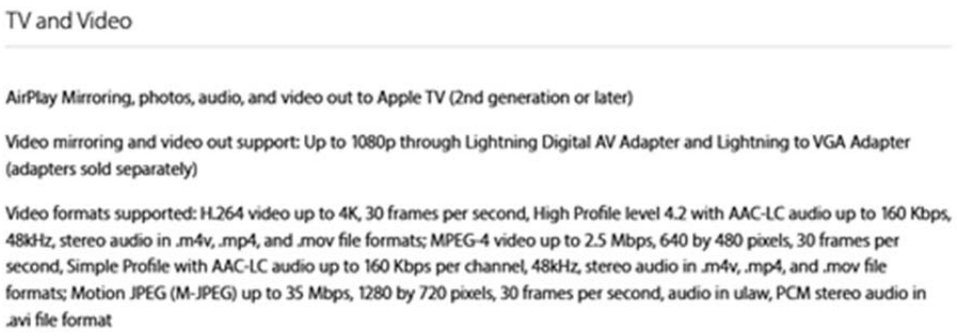
Source: H.264 Standard at § 8.2.5.2

91. Thus, as described above the Accused Products, including the Apple iPhone 7, infringe one or more claims of the ’005 patent, including claim 9.

92. Apple provides instruction manuals that instruct the users of the Accused Products to use the Accused Products in a manner that infringes the ’005 patent. In addition, Apple advertises the compatibility of the iPhone 7 with H.264 video file formats. *See* http://www.apple.com/iphone-7/specs/

F. Apple Makes, Imports, Uses, Sells, and/or Offers for Sale Products and Services that Infringe the ’211 Patent.

93. Each of the Accused Products is compliant with the H.264 Standard. For example, as shown below, Apple advertises that its iPhone 7 supports the H.264 video format.



TV and Video

AirPlay Mirroring, photos, audio, and video out to Apple TV (2nd generation or later)

Video mirroring and video out support: Up to 1080p through Lightning Digital AV Adapter and Lightning to VGA Adapter (adapters sold separately)

Video formats supported: H.264 video up to 4K, 30 frames per second, High Profile level 4.2 with AAC-LC audio up to 160 Kbps, 48kHz, stereo audio in .m4v, .mp4, and .mov file formats; MPEG-4 video up to 2.5 Mbps, 640 by 480 pixels, 30 frames per second, Simple Profile with AAC-LC audio up to 160 Kbps per channel, 48kHz, stereo audio in .m4v, .mp4, and .mov file formats; Motion JPEG (M-JPEG) up to 35 Mbps, 1280 by 720 pixels, 30 frames per second, audio in ulaw, PCM stereo audio in .avi file format

Source: http://www.apple.com/iphone-7/specs/.

94. Thus, for example and as shown below, the Accused Products infringe claim 50 of the '211 patent by virtue of their compatibility with and practice of the H.264 Standard. For example, the Accused Products comprise a decoder for performing motion compensated decoding of encoded video information, said decoder being arranged to derive prediction motion coefficients for blocks within a macroblock of a video frame being decoded from motion coefficients of at least one prediction block that is previously decoded macroblock or block within said video frame. As shown below, this functionality is described in the H.264 Standard, including but not limited to § 8.4.1.3 and Figure 6-14.

8.4.1.3 Derivation process for luma motion vector prediction

Inputs to this process are

- the macroblock partition index mbPartIdx,
- the sub-macroblock partition index subMbPartIdx,
- the reference index of the current partition refIdxLX (with X being 0 or 1),
- the variable currSubMbType.

Output of this process is the prediction mvpLX of the motion vector mvLX (with X being 0 or 1).

The derivation process for the neighbouring blocks for motion data in subclause 8.4.1.3.2 is invoked with mbPartIdx, subMbPartIdx, currSubMbType, and listSuffixFlag = X (with X being 0 or 1 for refIdxLX being refIdxL0 or refIdxL1, respectively) as the input and with mbAddrN\mbPartIdxN\subMbPartIdxN, reference indices refIdxLXN and the motion vectors mvLXN with N being replaced by A, B, or C as the output.

The derivation process for median luma motion vector prediction in subclause 8.4.1.3.1 is invoked with mbAddrN\mbPartIdxN\subMbPartIdxN, mvLXN, refIdxLXN with N being replaced by A, B, or C and refIdxLX as the input and mvpLX as the output, unless one of the following is true.

- MbPartWidth(mb_type) is equal to 16, MbPartHeight(mb_type) is equal to 8, mbPartIdx is equal to 0, and refIdxLXB is equal to refIdxLX,

  mvpLX = mvLXB (8-200)

- MbPartWidth(mb_type) is equal to 16, MbPartHeight(mb_type) is equal to 8, mbPartIdx is equal to 1, and refIdxLXA is equal to refIdxLX,

  mvpLX = mvLXA (8-201)

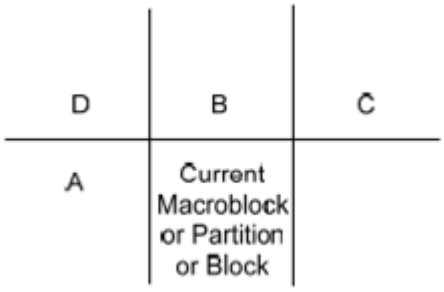
- MbPartWidth(mb_type) is equal to 8, MbPartHeight(mb_type) is equal to 16, mbPartIdx is equal to 0, and refIdxLXA is equal to refIdxLX,

  mvpLX = mvLXA (8-202)

- MbPartWidth(mb_type) is equal to 8, MbPartHeight(mb_type) is equal to 16, mbPartIdx is equal to 1, and refIdxLXC is equal to refIdxLX,

  mvpLX = mvLXC (8-203)

Source: H.264 Standard at § 8.4.1.3.



Source: H.264 Standard at Figure 6-14.

95. Further, the Accused Products comprise means for defining a certain number of available macroblock segmentations that specify possible ways in which a macroblock can be segmented into blocks and means for specifying at least one available prediction method for each available macroblock segmentation, thereby providing a certain finite number of available macroblock-segmentation-prediction-method pairs, each prediction method defining a method for deriving prediction motion coefficients for blocks within a macroblock using motion coefficients of at least one prediction block. As shown below, this structure and functionality is described in the H.264 Standard, including but not limited to § 8.4.1.3 and Figures 6-9 and 8-3.

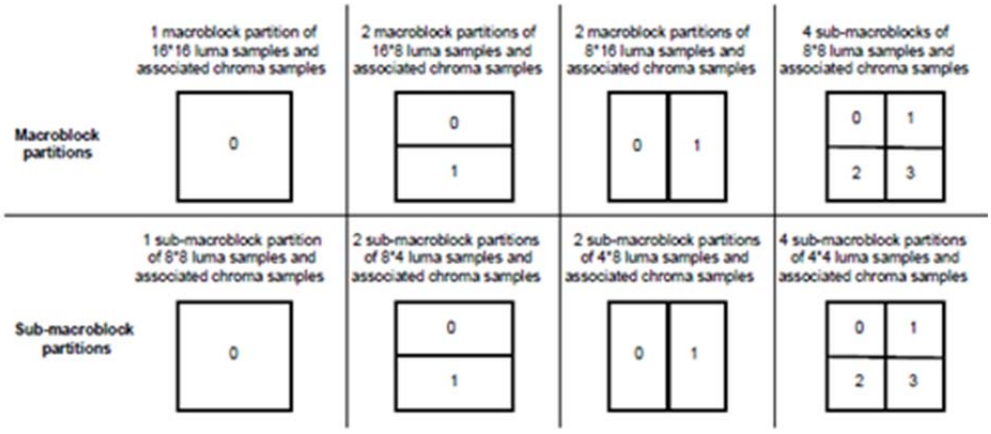
– MbPartWidth(mb_type) is equal to 16, MbPartHeight(mb_type) is equal to 8, mbPartIdx is equal to 0, and refIdxLXB is equal to refIdxLX,

$$mvpLX = mvLXB \quad (8\text{-}200)$$

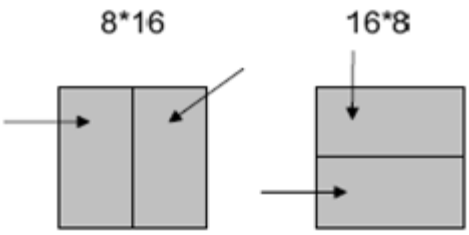
– MbPartWidth(mb_type) is equal to 16, MbPartHeight(mb_type) is equal to 8, mbPartIdx is equal to 1, and refIdxLXA is equal to refIdxLX,

$$mvpLX = mvLXA \quad (8\text{-}201)$$

Source: H.264 Standard at § 8.4.1.3.



Source: H.264 Standard at Figure 6-9.



Source: H.264 Standard at Figure 8-3.

96. The Accused Products also comprise input means for receiving information indicating at least the macroblock segmentation selected for a macroblock. As shown below, this structure and functionality is described in the H.264 Standard, including but not limited to §§ 8.4.1.3 and 7.4.5 and Table 7-13.

Table 7-13 – Macroblock type values 0 to 4 for P and SP slices

mb_type	Name of mb_type	NumMbPart (mb_type)	MbPartPredMode (mb_type, 0)	MbPartPredMode (mb_type, 1)	MbPartWidth (mb_type)	MbPartHeight (mb_type)
0	P_L0_16x16	1	Pred_L0	na	16	16
1	P_L0_L0_16x8	2	Pred_L0	Pred_L0	16	8
2	P_L0_L0_8x16	2	Pred_L0	Pred_L0	8	16
3	P_8x8	4	na	na	8	8
4	P_8x8ref0	4	na	na	8	8
inferred	P_Skip	1	Pred_L0	na	16	16

Source: H.264 Standard at Table 7-13.

97. The Accused Products also comprise means for determining the prediction-method relating to the segmentation of the macroblock with reference to the defined macroblock-segmentation—prediction-method pairs and means for producing prediction motion coefficients

for blocks within said macroblock using the determined prediction method. As shown below, this structure and functionality is described in the H.264 Standard, including but not limited to §§ 8.4.1.3 and 8.4.1.3.1.

8.4.1.3 Derivation process for luma motion vector prediction

Inputs to this process are

- the macroblock partition index mbPartIdx,
- the sub-macroblock partition index subMbPartIdx,
- the reference index of the current partition refIdxLX (with X being 0 or 1),
- the variable currSubMbType.

Output of this process is the prediction mvpLX of the motion vector mvLX (with X being 0 or 1).

The derivation process for the neighbouring blocks for motion data in subclause 8.4.1.3.2 is invoked with mbPartIdx, subMbPartIdx, currSubMbType, and listSuffixFlag = X (with X being 0 or 1 for refIdxLX being refIdxL0 or refIdxL1, respectively) as the input and with mbAddrN\mbPartIdxN\subMbPartIdxN, reference indices refIdxLXN and the motion vectors mvLXN with N being replaced by A, B, or C as the output.

The derivation process for median luma motion vector prediction in subclause 8.4.1.3.1 is invoked with mbAddrN\mbPartIdxN\subMbPartIdxN, mvLXN, refIdxLXN with N being replaced by A, B, or C and refIdxLX as the input and mvpLX as the output, unless one of the following is true.

- MbPartWidth(mb_type) is equal to 16, MbPartHeight(mb_type) is equal to 8, mbPartIdx is equal to 0, and refIdxLXB is equal to refIdxLX,

  mvpLX = mvLXB (8-200)

- MbPartWidth(mb_type) is equal to 16, MbPartHeight(mb_type) is equal to 8, mbPartIdx is equal to 1, and refIdxLXA is equal to refIdxLX,

  mvpLX = mvLXA (8-201)

- MbPartWidth(mb_type) is equal to 8, MbPartHeight(mb_type) is equal to 16, mbPartIdx is equal to 1, and refIdxLXC is equal to refIdxLX,

  mvpLX = mvLXC (8-203)

Source: H.264 Standard at § 8.4.1.3.

98. Thus, as described above the Accused Products, including the Apple iPhone 7, infringe one or more claims of the ’211 patent, including claim 50.

99. Apple provides instruction manuals that instruct the users of the Accused Products to use the Accused Products in a manner that infringes the ’211 patent. For example, Apple advertises the compatibility of the iPhone 7 with H.264 video file formats. *See* http://www.apple.com/iphone-7/specs/.

G. Apple Makes, Imports, Uses, Sells, and/or Offers for Sale Products and Services that Infringe the '701 Patent.

100. Each of the Accused Products is compliant with the H.264 Standard. For example, as shown below, Apple advertises that its iPhone 7 supports the H.264 video format and recording video.

Video Recording	4K video recording at 30 fps 1080p HD video recording at 30 fps or 60 fps 720p HD video recording at 30 fps Optical image stabilization for video (iPhone 6s Plus only) True Tone flash Slo-mo video support for 1080p at 120 fps and 720p at 240 fps Time-lapse video with stabilization Cinematic video stabilization (1080p and 720p) Continuous autofocus video Noise reduction Take 8-megapixel still photos while recording 4K video Playback zoom 3x digital zoom Face detection Video geotagging
TV and Video	AirPlay Mirroring, photos, audio, and video out to Apple TV (2nd generation or later) Video mirroring and video out support: Up to 1080p through Lightning Digital AV Adapter and Lightning to VGA Adapter (adapters sold separately) Video formats supported: H.264 video up to 4K, 30 frames per second, High Profile level 4.2 with AAC-LC audio up to 160 Kbps, 48kHz, stereo audio in .m4v, .mp4, and .mov file formats; MPEG-4 video up to 2.5 Mbps, 640 by 480 pixels, 30 frames per second, Simple Profile with AAC-LC audio up to 160 Kbps per channel, 48kHz, stereo audio in .m4v, .mp4, and .mov file formats; Motion JPEG (M-JPEG) up to 35 Mbps, 1280 by 720 pixels, 30 frames per second, audio in ulaw, PCM stereo audio in .avi file format

Source: http://www.apple.com/iphone-7/specs/.

101. Thus, for example and as shown below, the Accused Products infringe claim 18 of the '701 patent by virtue of their compatibility with and practice of the H.264 Standard. For

example, the Accused Products comprise a system for image coding comprising an input for receiving an image as a plurality of blocks having a plurality of pixels, each pixel having a pixel value. As shown below, this functionality is described in the H.264 Standard, including but not limited to §§ 3.53 and 6.3.

3.53 **frame**: A *frame* contains an array of *luma* samples and two corresponding arrays of *chroma* samples. A *frame* consists of two *fields*, a *top field* and a *bottom field*.

Source: H.264 Standard at § 3.53.

6.3 Spatial subdivision of pictures and slices

This subclause specifies how a picture is partitioned into slices and macroblocks. Pictures are divided into slices. A slice is a sequence of macroblocks, or, when macroblock-adaptive frame/field decoding is in use, a sequence of macroblock pairs.

Each macroblock is comprised of one 16x16 luma array and, when the video format is not monochrome, two corresponding chroma sample arrays. When macroblock-adaptive frame/field decoding is not in use, each macroblock represents a spatial rectangular region of the picture. For example, a picture may be divided into two slices as shown in Figure 6-7.

Source: H.264 Standard at § 6.3.

102. Further, the Accused Products comprise a transform coder for performing a transform coding operation on a block of pixels to produce a corresponding block of transform coefficient values As shown below, this structure and functionality is described in the H.264 Standard, including but not limited to §§ 8.5.1 and 8.5.10.

8.5.1 Specification of transform decoding process for 4x4 luma residual blocks

This specification applies when transform_size_8x8_flag is equal to 0.

When the current macroblock prediction mode is not equal to Intra_16x16, the variable LumaLevel contains the levels for the luma transform coefficients. For a 4x4 luma block indexed by luma4x4BlkIdx = 0..15, the following ordered steps are specified.

1. The inverse transform coefficient scanning process as described in subclause 8.5.5 is invoked with LumaLevel[luma4x4BlkIdx] as the input and the two-dimensional array c as the output.
2. The scaling and transformation process for residual 4x4 blocks as specified in subclause 8.5.10 is invoked with c as the input and r as the output.

Source: H.264 Standard at § 8.5.1.

8.5.10 Scaling and transformation process for residual 4x4 blocks

Input to this process is a 4x4 array c with elements c_{ij} which is either an array relating to a residual block of the luma component or an array relating to a residual block of a chroma component.

Outputs of this process are residual sample values as 4x4 array r with elements r_{ij}.

.
.
.

After performing both the one-dimensional horizontal and the one-dimensional vertical inverse transforms to produce an array of transformed samples, the final constructed residual sample values is derived as

$$r_{ij} = (h_{ij} + 2^5) >> 6 \quad \text{with} \quad i, j = 0..3 \qquad (8\text{-}357)$$

Source: H.264 Standard at § 8.5.10.

103. Further, the Accused Products comprise a scanner for scanning the block of transform coefficient values in a given scanning order to produce a scanned array of coefficient values arranged according to the scanning order. As shown below, this structure and functionality is described in the H.264 Standard, including but not limited to §§ 8.5.1 and 8.5.5 and Figure 8-8.

8.5.1 Specification of transform decoding process for 4x4 luma residual blocks

This specification applies when transform_size_8x8_flag is equal to 0.

When the current macroblock prediction mode is not equal to Intra_16x16, the variable LumaLevel contains the levels for the luma transform coefficients. For a 4x4 luma block indexed by luma4x4BlkIdx = 0..15, the following ordered steps are specified.

1. The inverse transform coefficient scanning process as described in subclause 8.5.5 is invoked with LumaLevel[luma4x4BlkIdx] as the input and the two-dimensional array c as the output.
2. The scaling and transformation process for residual 4x4 blocks as specified in subclause 8.5.10 is invoked with c as the input and r as the output.

Source: H.264 Standard at § 8.5.1.

8.5.5 Inverse scanning process for transform coefficients

Input to this process is a list of 16 values.

Output of this process is a variable c containing a two-dimensional array of 4x4 values. In the case of transform coefficients, these 4x4 values represent levels assigned to locations in the transform block. In the case of applying the inverse scanning process to a scaling list, the output variable c contains a two-dimensional array representing a 4x4 scaling matrix.

The inverse scanning process for transform coefficients maps the sequence of transform coefficient levels to the transform coefficient level positions. Table 8-13 specifies the two mappings: inverse zig-zag scan and inverse field scan. The inverse zig-zag scan is used for transform coefficients in frame macroblocks and the inverse field scan is used for transform coefficients in field macroblocks.

The inverse scanning process for scaling lists maps the sequence of scaling list entries to the positions in the corresponding scaling matrix. For this mapping, the inverse zig-zag scan is used.

Figure 8-8 illustrates the scans.

Source: H.264 Standard at § 8.5.5.

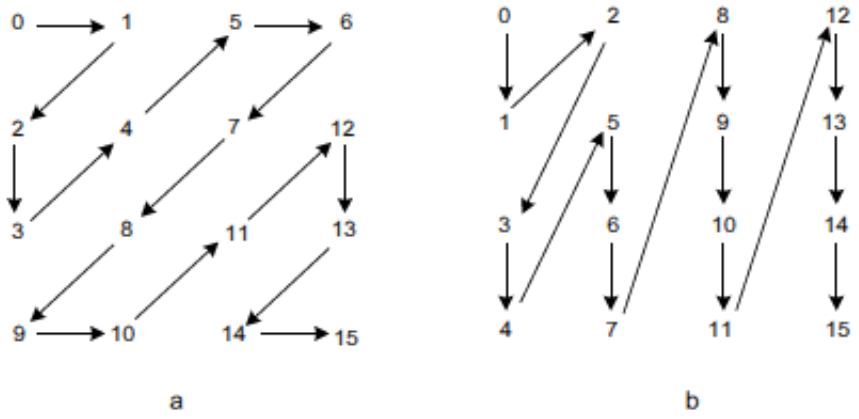


Figure 8-8 – 4x4 block scans. (a) Zig-zag scan. (b) Field scan (informative)

Source: H.264 Standard at Figure 8-8.

104. Further, the Accused Products comprise a run-level coder for representing the coefficient values in the scanned array by a plurality of number pairs, said number pairs having a first number and a second number. As shown below, this structure and functionality is described in the H.264 Standard, including but not limited to §§ 7.4.5.3.2 and 9.3.

significant_coeff_flag[i **]** specifies whether the transform coefficient level at scanning position i is non-zero as follows.

- If significant_coeff_flag[i] is equal to 0, the transform coefficient level at scanning position i is set equal to 0;
- Otherwise (significant_coeff_flag[i] is equal to 1), the transform coefficient level at scanning position i has a non-zero value.

.

.
.

coeff_abs_level_minus1[i] is the absolute value of a transform coefficient level minus 1. The value of coeff_abs_level_minus1 is constrained by the limits in subclause 8.5.

Source: H.264 Standard at § 7.4.5.3.2.

9.3 CABAC parsing process for slice data

This process is invoked when parsing syntax elements with descriptor ae(v) in subclauses 7.3.4 and 7.3.5 when entropy_coding_mode_flag is equal to 1.

Source: H.264 Standard at § 9.3.

105. Further, the Accused Products comprise a context-based coder for assigning the first numbers to one of a plurality of contexts representative of the first numbers and operative to assign the first number of a first number pair to a context at least partly in dependence on a first number of a second number pair. As shown below, this structure and functionality is described in the H.264 Standard, including but not limited to § 9.3.3.1.3.

9.3.3.1.3 Assignment process of ctxIdxInc for syntax elements significant_coeff_flag, last_significant_coeff_flag, and coeff_abs_level_minus1

Inputs to this process are ctxIdxOffset and binIdx.

Output of this process is ctxIdxInc.

The assignment process of ctxIdxInc for syntax elements significant_coeff_flag, last_significant_coeff_flag, and coeff_abs_level_minus1 as well as for coded_block_flag depends on categories of different blocks denoted by the variable ctxBlockCat. The specification of these block categories is given in Table 9-33.

.
.
.

Let numDecodAbsLevelEq1 denotes the accumulated number of decoded transform coefficient levels with absolute value equal to 1, and let numDecodAbsLevelGt1 denotes the accumulated number of decoded transform coefficient levels with absolute value greater than 1. Both numbers are related to the same transform coefficient block, where the current decoding process takes place. Then, for decoding of coeff_abs_level_minus1, ctxIdxInc for coeff_abs_level_minus1 is specified depending on binIdx as follows.

– If binIdx is equal to 0, ctxIdxInc is derived by

ctxIdxInc = ((numDecodAbsLevelGt1 != 0) ? 0: Min(4, 1 + numDecodAbsLevelEq1)) (9-17)

– Otherwise (binIdx is greater than 0), ctxIdxInc is derived by

ctxIdxInc = 5 + Min(4 – (ctxBlockCat == 3), numDecodAbsLevelGt1) (9-18)

Source: H.264 Standard at § 9.3.3.1.3.

106. Thus, as described above the Accused Products, including the Apple iPhone 7, infringe one or more claims of the '701 patent, including claim 18.

107. Apple provides instruction manuals that instruct the users of the Accused Products to use the Accused Products in a manner that infringes the '701 patent. For example, Apple advertises the compatibility of the iPhone 7 with H.264 video file formats and its ability to record video. *See* http://www.apple.com/iphone-7/specs/.

H. Apple Makes, Imports, Uses, Sells, and/or Offers for Sale Products and Services that Infringe the '974 Patent.

108. Each of the Accused Products encodes video files capable of being decoded according to the H.264 technical standard. For example, as shown below, Apple advertises that its iPhone 7 supports the H.264 video format.

TV and Video	AirPlay Mirroring, photos, audio, and video out to Apple TV (2nd generation or later) Video mirroring and video out support: Up to 1080p through Lightning Digital AV Adapter and Lightning to VGA Adapter (adapters sold separately) Video formats supported: H.264 video up to 4K, 30 frames per second, High Profile level 4.2 with AAC-LC audio up to 160 Kbps, 48kHz, stereo audio or Dolby Audio up to 1008 Kbps, 48kHz, stereo or multichannel audio, in .m4v, .mp4, and .mov file formats; MPEG-4 video up to 2.5 Mbps, 640 by 480 pixels, 30 frames per second, Simple Profile with AAC-LC audio up to 160 Kbps per channel, 48kHz, stereo audio or Dolby Audio up to 1008 Kbps, 48kHz, stereo or multichannel audio, in .m4v, .mp4, and .mov file formats; Motion JPEG (M-JPEG) up to 35 Mbps, 1280 by 720 pixels, 30 frames per second, audio in ulaw, PCM stereo audio in .avi file format

Source: http://www.apple.com/iphone-7/specs/

109. Thus, for example and as shown below, the Accused Products infringe claim 15 of the '974 patent by virtue of their encoding of video files capable of being decoded according to the H.264 Standard. For example, the Accused Products comprise an apparatus for use in a block transform-based coding system for processing one or more block transform coefficients associated with at least one block of visual data. This functionality is described in the H.264 Standard, including but not limited to §§ 6.4.2 and 8.5

3.15 **block**: An MxN (M-column by N-row) array of samples, or an MxN array of *transform coefficients*.

3.75 **macroblock**: A 16x16 *block* of *luma* samples and two corresponding *blocks* of *chroma* samples. The division of a *slice* or a *macroblock pair* into macroblocks is a *partitioning*.

Source: H.264 Standard at §§ 3.15 and 3.75.

8.5.5 Inverse scanning process for transform coefficients

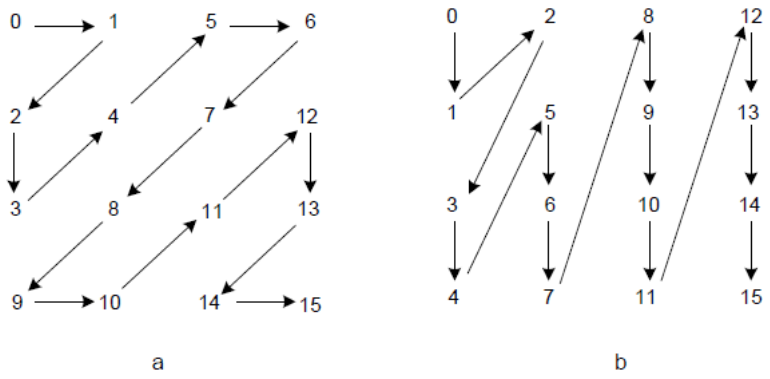
Input to this process is a list of 16 values.

Output of this process is a variable c containing a two-dimensional array of 4x4 values. In the case of transform coefficients, these 4x4 values represent levels assigned to locations in the transform block. In the case of applying the inverse scanning process to a scaling list, the output variable c contains a two-dimensional array representing a 4x4 scaling matrix.

The inverse scanning process for transform coefficients maps the sequence of transform coefficient levels to the transform coefficient level positions. Table 8-13 specifies the two mappings: inverse zig-zag scan and inverse field scan. The inverse zig-zag scan is used for transform coefficients in frame macroblocks and the inverse field scan is used for transform coefficients in field macroblocks.

The inverse scanning process for scaling lists maps the sequence of scaling list entries to the positions in the corresponding scaling matrix. For this mapping, the inverse zig-zag scan is used.

Figure 8-8 illustrates the scans.



Source: H.264 Standard at § 8.5.5.

8.5.6 Inverse scanning process for 8x8 luma transform coefficients

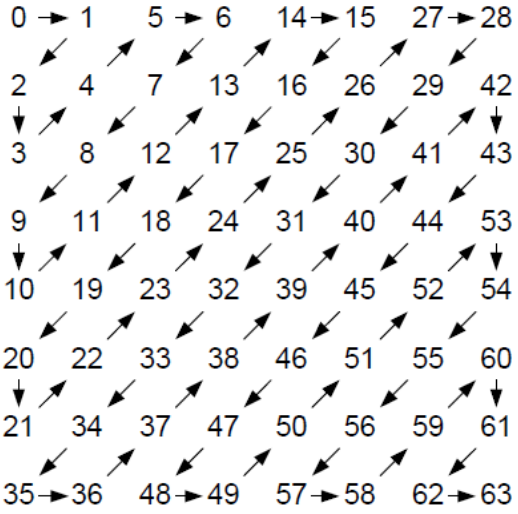
Input to this process is a list of 64 values.

Output of this process is a variable c containing a two-dimensional array of 8x8 values. In the case of transform coefficients, these 8x8 values represent levels assigned to locations in the transform block. In the case of applying the inverse scanning process to a scaling list, the output variable c contains a two-dimensional array representing an 8x8 scaling matrix.

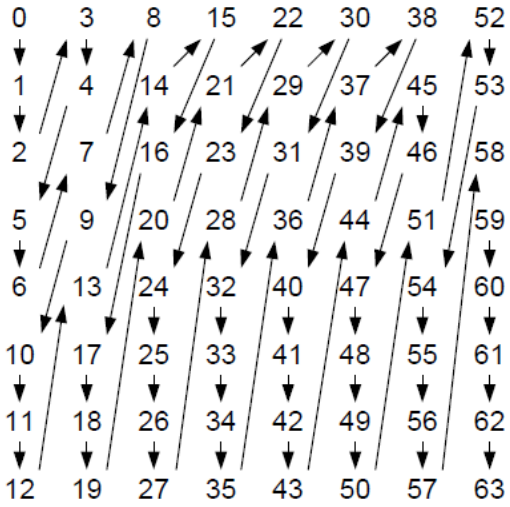
The inverse scanning process for transform coefficients maps the sequence of transform coefficient levels to the transform coefficient level positions. Table 8-14 specifies the two mappings: inverse 8x8 zig-zag scan and inverse 8x8 field scan. The inverse 8x8 zig-zag scan is used for transform coefficients in frame macroblocks and the inverse 8x8 field scan is used for transform coefficients in field macroblocks.

The inverse scanning process for scaling lists maps the sequence of scaling list entries to the positions in the corresponding scaling matrix. For this mapping, the inverse zig-zag scan is used.

Figure 8-9 illustrates the scans.



a



b

Source: H.264 Standard at § 8.5.6.

110. The Accused Products also include at least one processing device operative to: (i) identify one or more previously reconstructed block transform coefficients associated with the visual data; and (ii) compute a context selection value for use in processing a block transform coefficient associated with the at least one block, the context selection value being based on the one or more previously reconstructed block transform coefficients, wherein the apparatus is included in a video encoder of the system and the processing operation includes encoding the

block transform coefficient. This functionality is described in the H.264 Standard, including but not limited to §§ 6.4 and 9.2.

- The variables blkA and blkB are derived as follows.
  - If the CAVLC parsing process is invoked for Intra16x16DCLevel, Intra16x16ACLevel, or LumaLevel, the process specified in subclause 6.4.8.3 is invoked with luma4x4BlkIdx as the input, and the output is assigned to mbAddrA, mbAddrB, luma4x4BlkIdxA, and luma4x4BlkIdxB. The 4x4 luma block specified by mbAddrA\luma4x4BlkIdxA is assigned to blkA, and the 4x4 luma block specified by mbAddrB\luma4x4BlkIdxB is assigned to blkB.
  - Otherwise (the CAVLC parsing process is invoked for ChromaACLevel), the process specified in subclause 6.4.8.4 is invoked with chroma4x4BlkIdx as input, and the output is assigned to mbAddrA, mbAddrB, chroma4x4BlkIdxA, and chroma4x4BlkIdxB. The 4x4 chroma block specified by mbAddrA\iCbCr\chroma4x4BlkIdxA is assigned to blkA, and the 4x4 chroma block specified by mbAddrB\iCbCr\chroma4x4BlkIdxB is assigned to blkB.

Source: H.264 Standard at § 9.2.1

- Given the values of nA and nB, the variable nC is derived as follows.
  - If both mbAddrA and mbAddrB are available, the variable nC is set equal to (nA + nB + 1) >> 1.
  - Otherwise (mbAddrA is not available or mbAddrB is not available), the variable nC is set equal to nA + nB.

The value of TotalCoeff(coeff_token) resulting from decoding coeff_token shall be in the range of 0 to maxNumCoeff, inclusive.

Source: H.264 Standard at § 9.2.1

7.3.5.3.1 Residual block CAVLC syntax

residual_block_cavlc(coeffLevel, maxNumCoeff) {	C	Descriptor
for(i = 0; i < maxNumCoeff; i++)		
coeffLevel[i] = 0		
coeff_token	3 | 4	ce(v)
if(TotalCoeff(coeff_token) > 0) {		
if(TotalCoeff(coeff_token) > 10 && TrailingOnes(coeff_token) < 3)		
suffixLength = 1		
else		
suffixLength = 0		
for(i = 0; i < TotalCoeff(coeff_token); i++)		
if(i < TrailingOnes(coeff_token)) {		
trailing_ones_sign_flag	3 | 4	u(1)
level[i] = 1 – 2 * trailing_ones_sign_flag		
} else {		
level_prefix	3 | 4	ce(v)
levelCode = (Min(15, level_prefix) << suffixLength)		
if(suffixLength > 0 || level_prefix >= 14) {		
level_suffix	3 | 4	u(v)
levelCode += level_suffix		
}		
if(level_prefix >= 15 && suffixLength == 0)		
levelCode += 15		
if(level_prefix >= 16)		
levelCode += (1 << (level_prefix – 3)) – 4096		
if(i == TrailingOnes(coeff_token) && TrailingOnes(coeff_token) < 3)		
levelCode += 2		
if(levelCode % 2 == 0)		
level[i] = (levelCode + 2) >> 1		
else		
level[i] = (–levelCode – 1) >> 1		
if(suffixLength == 0)		
suffixLength = 1		
if(Abs(level[i]) > (3 << (suffixLength – 1)) && suffixLength < 6)		
suffixLength++		
}		
if(TotalCoeff(coeff_token) < maxNumCoeff) {		
total_zeros	3 | 4	ce(v)
zerosLeft = total_zeros		
} else		
zerosLeft = 0		
for(i = 0; i < TotalCoeff(coeff_token) – 1; i++) {		
if(zerosLeft > 0) {		
run_before	3 | 4	ce(v)
run[i] = run_before		
} else		
run[i] = 0		

zerosLeft = zerosLeft – run[i]		
}		
run[TotalCoeff(coeff_token) – 1] = zerosLeft		
coeffNum = -1		
for(i = TotalCoeff(coeff_token) – 1; i >= 0; i--) {		
coeffNum += run[i] + 1		
coeffLevel[coeffNum] = level[i]		
}		
}		
}		

Source: H.264 Standard at §7.3.5.3.1

111. This functionality is performed by the H.264 encoder implemented in the Accused Products. Thus, as described above the Accused Products, including the Apple iPhone 7, infringe one or more claims of the '974 patent, including claim 15.

112. Apple provides instruction manuals that instruct the users of the Accused Products to use the Accused Products in a manner that infringes the '974 patent. In addition, Apple advertises the compatibility of the iPhone 7 with H.264 video file formats. *See* http://www.apple.com/iphone-7/specs/

COUNT I: PATENT INFRINGEMENT OF THE '808 PATENT

113. Nokia incorporates by reference the preceding paragraphs as though fully set forth herein.

114. Apple infringes, contributes to the infringement of, and/or induces infringement of the '808 Patent by making, using, selling, offering for sale, and/or importing into the United States products and/or methods covered by one or more claims of the '808 Patent including, but not limited to, at least the Apple Accused Products. The accused devices that infringe one or more claims of the '808 Patent include, but are not limited to, at least the Apple Accused Products.

115. The Apple Accused Products directly infringe one or more claims of the '808 Patent. Apple makes, uses, sells, offers for sale, and/or imports, in this District and elsewhere in the United States these devices and thus directly infringes the '808 Patent.

116. Apple has had knowledge and notice of the '808 Patent at least as of August 31, 2014, by virtue of Nokia presenting the '808 Patent to Apple. Apple has been involved in licensing discussions with Nokia regarding Nokia's patent portfolio, which includes the '808 Patent.

117. Apple indirectly infringes the '808 Patent, as provided in 35 U.S.C. § 271(b), by inducing infringement by others, such as Apple's customers and end-users, in this District and elsewhere in the United States. For example, Apple's customers and end-users directly infringe through their use of the inventions claimed in the '808 Patent. Apple induces this direct infringement through its affirmative acts of manufacturing, selling, distributing, and/or otherwise making available the Apple Accused Products, and providing instructions, documentation, and other information to customers and end users suggesting they use the Apple Accused Products in an infringing manner, including in-store technical support, online technical support, marketing, product manuals, advertisements, online documentation, developer information, and API documentation. As a result of Apple's inducement, Apple's customers and end users use the Apple Accused Products in the way Apple intends and directly infringe the '808 Patent. Apple has performed and continues to perform these affirmative acts with knowledge of the '808 Patent and with the intent, or willful blindness, that the induced acts directly infringe the '808 Patent.

118. Apple also indirectly infringes the '808 Patent, as provided by 35 U.S.C. § 271(c), by contributing to direct infringement committed by others, such as customers and end-users, in this District and elsewhere in the United States. Apple's affirmative acts of selling and offering

to sell, in this District and elsewhere in the United States, the Apple Accused Products and causing the Apple Accused Products to be manufactured, used, sold, and offered for sale contribute to Apple's customers and end-users use of the Apple Accused Products, such that the '808 Patent is directly infringed. The accused components within the Apple Accused Products are material to the invention of the '808 Patent, are not staple articles or commodities of commerce, have no substantial non-infringing uses, and are known by Apple to be especially made or especially adapted for use in infringement of the '808 Patent. Apple has performed and continues to perform these affirmative acts with knowledge of the '808 Patent and with intent, or willful blindness, that they cause the direct infringement of the '808 Patent.

119. Apple's infringement of the '808 Patent has damaged and will continue to damage Nokia.

COUNT II: PATENT INFRINGEMENT OF THE '469 PATENT

120. Nokia incorporates by reference the preceding paragraphs as though fully set forth herein.

121. Apple infringes, contributes to the infringement of, and/or induces infringement of the '469 Patent by making, using, selling, offering for sale, and/or importing into the United States products and/or methods covered by one or more claims of the '469 Patent including, but not limited to, at least the Apple Accused Products. The accused devices that infringe one or more claims of the '469 Patent include, but are not limited to, at least the Apple Accused Products.

122. The Apple Accused Products directly infringe one or more claims of the '469 Patent. Apple makes, uses, sells, offers for sale, and/or imports, in this District and elsewhere in the United States these devices and thus directly infringes the '469 Patent.

123. Apple has had knowledge and notice of the '469 Patent at least as of August 31, 2014, by virtue of Nokia presenting the '469 Patent to Apple. Apple has been involved in licensing discussions with Nokia regarding Nokia's patent portfolio, which includes the '469 Patent.

124. Apple indirectly infringes the '469 Patent, as provided in 35 U.S.C. § 271(b), by inducing infringement by others, such as Apple's customers and end-users, in this District and elsewhere in the United States. For example, Apple's customers and end-users directly infringe through their use of the inventions claimed in the '469 Patent. Apple induces this direct infringement through its affirmative acts of manufacturing, selling, distributing, and/or otherwise making available the Apple Accused Products, and providing instructions, documentation, and other information to customers and end users suggesting they use the Apple Accused Products in an infringing manner, including in-store technical support, online technical support, marketing, product manuals, advertisements, online documentation, developer information, and API documentation. As a result of Apple's inducement, Apple's customers and end users use the Apple Accused Products in the way Apple intends and directly infringe the '469 Patent. Apple has performed and continues to perform these affirmative acts with knowledge of the '469 Patent and with the intent, or willful blindness, that the induced acts directly infringe the '469 Patent.

125. Apple also indirectly infringes the '469 Patent, as provided by 35 U.S.C. § 271(c), by contributing to direct infringement committed by others, such as customers and end-users, in this District and elsewhere in the United States. Apple's affirmative acts of selling and offering to sell, in this District and elsewhere in the United States, the Apple Accused Products and causing the Apple Accused Products to be manufactured, used, sold, and offered for sale contribute to Apple's customers and end-users use of the Apple Accused Products, such that the

'469 Patent is directly infringed. The accused components within the Apple Accused Products are material to the invention of the '469 Patent, are not staple articles or commodities of commerce, have no substantial non-infringing uses, and are known by Apple to be especially made or especially adapted for use in infringement of the '469 Patent. Apple has performed and continues to perform these affirmative acts with knowledge of the '469 Patent and with intent, or willful blindness, that they cause the direct infringement of the '469 Patent.

126. Apple's infringement of the '469 Patent has damaged and will continue to damage Nokia.

COUNT III: PATENT INFRINGEMENT OF THE '273 PATENT

127. Nokia incorporates by reference the preceding paragraphs as though fully set forth herein.

128. Apple infringes, contributes to the infringement of, and/or induces infringement of the '273 Patent by making, using, selling, offering for sale, and/or importing into the United States products and/or methods covered by one or more claims of the '273 Patent including, but not limited to, at least the Apple Accused Products. The accused devices that infringe one or more claims of the '273 Patent include, but are not limited to, at least the Apple Accused Products.

129. The Apple Accused Products directly infringe one or more claims of the '273 Patent. Apple makes, uses, sells, offers for sale, and/or imports, in this District and elsewhere in the United States these devices and thus directly infringes the '273 Patent.

130. Apple has had knowledge and notice of the '273 Patent at least as of August 31, 2014, by virtue of Nokia presenting the '273 Patent to Apple. Apple has been involved in

licensing discussions with Nokia regarding Nokia's patent portfolio, which includes the '273 Patent.

131. Apple indirectly infringes the '273 Patent, as provided in 35 U.S.C. § 271(b), by inducing infringement by others, such as Apple's customers and end-users, in this District and elsewhere in the United States. For example, Apple's customers and end-users directly infringe through their use of the inventions claimed in the '273 Patent. Apple induces this direct infringement through its affirmative acts of manufacturing, selling, distributing, and/or otherwise making available the Apple Accused Products, and providing instructions, documentation, and other information to customers and end users suggesting they use the Apple Accused Products in an infringing manner, including in-store technical support, online technical support, marketing, product manuals, advertisements, online documentation, developer information, and API documentation. As a result of Apple's inducement, Apple's customers and end users use the Apple Accused Products in the way Apple intends and directly infringe the '273 Patent. Apple has performed and continues to perform these affirmative acts with knowledge of the '273 Patent and with the intent, or willful blindness, that the induced acts directly infringe the '273 Patent.

132. Apple also indirectly infringes the '273 Patent, as provided by 35 U.S.C. § 271(c), by contributing to direct infringement committed by others, such as customers and end-users, in this District and elsewhere in the United States. Apple's affirmative acts of selling and offering to sell, in this District and elsewhere in the United States, the Apple Accused Products and causing the Apple Accused Products to be manufactured, used, sold, and offered for sale contribute to Apple's customers and end-users use of the Apple Accused Products, such that the '273 Patent is directly infringed. The accused components within the Apple Accused Products are material to the invention of the '273 Patent, are not staple articles or commodities of

commerce, have no substantial non-infringing uses, and are known by Apple to be especially made or especially adapted for use in infringement of the ’273 Patent. Apple has performed and continues to perform these affirmative acts with knowledge of the ’273 Patent and with intent, or willful blindness, that they cause the direct infringement of the ’273 Patent.

133. Apple’s infringement of the ’273 Patent has damaged and will continue to damage Nokia.

COUNT IV: PATENT INFRINGEMENT OF THE ’764 PATENT

134. Nokia incorporates by reference the preceding paragraphs as though fully set forth herein.

135. Apple infringes, contributes to the infringement of, and/or induces infringement of the ’764 Patent by making, using, selling, offering for sale, and/or importing into the United States products and/or methods covered by one or more claims of the ’764 Patent including, but not limited to, at least the Apple Accused Products. The accused devices that infringe one or more claims of the ’764 Patent include, but are not limited to, at least the Apple Accused Products.

136. The Apple Accused Products directly infringe one or more claims of the ’764 Patent. Apple makes, uses, sells, offers for sale, and/or imports, in this District and elsewhere in the United States these devices and thus directly infringes the ’764 Patent.

137. Apple has had knowledge and notice of the ’764 Patent at least as of August 31, 2014, by virtue of Nokia presenting the ’764 Patent to Apple. Apple has been involved in licensing discussions with Nokia regarding Nokia’s patent portfolio, which includes the ’764 Patent.

138. Apple indirectly infringes the '764 Patent, as provided in 35 U.S.C. § 271(b), by inducing infringement by others, such as Apple's customers and end-users, in this District and elsewhere in the United States. For example, Apple's customers and end-users directly infringe through their use of the inventions claimed in the '764 Patent. Apple induces this direct infringement through its affirmative acts of manufacturing, selling, distributing, and/or otherwise making available the Apple Accused Products, and providing instructions, documentation, and other information to customers and end users suggesting they use the Apple Accused Products in an infringing manner, including in-store technical support, online technical support, marketing, product manuals, advertisements, online documentation, developer information, and API documentation. As a result of Apple's inducement, Apple's customers and end users use the Apple Accused Products in the way Apple intends and directly infringe the '764 Patent. Apple has performed and continues to perform these affirmative acts with knowledge of the '764 Patent and with the intent, or willful blindness, that the induced acts directly infringe the '764 Patent.

139. Apple also indirectly infringes the '764 Patent, as provided by 35 U.S.C. § 271(c), by contributing to direct infringement committed by others, such as customers and end-users, in this District and elsewhere in the United States. Apple's affirmative acts of selling and offering to sell, in this District and elsewhere in the United States, the Apple Accused Products and causing the Apple Accused Products to be manufactured, used, sold, and offered for sale contribute to Apple's customers and end-users use of the Apple Accused Products, such that the '764 Patent is directly infringed. The accused components within the Apple Accused Products are material to the invention of the '764 Patent, are not staple articles or commodities of commerce, have no substantial non-infringing uses, and are known by Apple to be especially made or especially adapted for use in infringement of the '764 Patent. Apple has performed and

continues to perform these affirmative acts with knowledge of the '764 Patent and with intent, or willful blindness, that they cause the direct infringement of the '764 Patent.

140. Apple's infringement of the '764 Patent has damaged and will continue to damage Nokia.

COUNT V: PATENT INFRINGEMENT OF THE '005 PATENT

141. Nokia incorporates by reference the preceding paragraphs as though fully set forth herein.

142. Apple infringes, contributes to the infringement of, and/or induces infringement of the '005 Patent by making, using, selling, offering for sale, and/or importing into the United States products and/or methods covered by one or more claims of the '005 Patent including, but not limited to, at least the Apple Accused Products. The accused devices that infringe one or more claims of the '005 Patent include, but are not limited to, at least the Apple Accused Products.

143. The Apple Accused Products directly infringe one or more claims of the '005 Patent. Apple makes, uses, sells, offers for sale, and/or imports, in this District and elsewhere in the United States these devices and thus directly infringes the '005 Patent.

144. Apple has had knowledge and notice of the '005 Patent at least as of August 31, 2014, by virtue of Nokia presenting the '005 Patent to Apple. Apple has been involved in licensing discussions with Nokia regarding Nokia's patent portfolio, which includes the '005 Patent.

145. Apple indirectly infringes the '005 Patent, as provided in 35 U.S.C. § 271(b), by inducing infringement by others, such as Apple's customers and end-users, in this District and elsewhere in the United States. For example, Apple's customers and end-users directly infringe

through their use of the inventions claimed in the '005 Patent. Apple induces this direct infringement through its affirmative acts of manufacturing, selling, distributing, and/or otherwise making available the Apple Accused Products, and providing instructions, documentation, and other information to customers and end users suggesting they use the Apple Accused Products in an infringing manner, including in-store technical support, online technical support, marketing, product manuals, advertisements, online documentation, developer information, and API documentation. As a result of Apple's inducement, Apple's customers and end users use the Apple Accused Products in the way Apple intends and directly infringe the '005 Patent. Apple has performed and continues to perform these affirmative acts with knowledge of the '005 Patent and with the intent, or willful blindness, that the induced acts directly infringe the '005 Patent.

146. Apple also indirectly infringes the '005 Patent, as provided by 35 U.S.C. § 271(c), by contributing to direct infringement committed by others, such as customers and end-users, in this District and elsewhere in the United States. Apple's affirmative acts of selling and offering to sell, in this District and elsewhere in the United States, the Apple Accused Products and causing the Apple Accused Products to be manufactured, used, sold, and offered for sale contribute to Apple's customers and end-users use of the Apple Accused Products, such that the '005 Patent is directly infringed. The accused components within the Apple Accused Products are material to the invention of the '005 Patent, are not staple articles or commodities of commerce, have no substantial non-infringing uses, and are known by Apple to be especially made or especially adapted for use in infringement of the '005 Patent. Apple has performed and continues to perform these affirmative acts with knowledge of the '005 Patent and with intent, or willful blindness, that they cause the direct infringement of the '005 Patent.

147. Apple's infringement of the '005 Patent has damaged and will continue to damage Nokia.

COUNT VI: PATENT INFRINGEMENT OF THE '211 PATENT

148. Nokia incorporates by reference the preceding paragraphs as though fully set forth herein.

149. Apple infringes, contributes to the infringement of, and/or induces infringement of the '211 Patent by making, using, selling, offering for sale, and/or importing into the United States products and/or methods covered by one or more claims of the '211 Patent including, but not limited to, at least the Apple Accused Products. The accused devices that infringe one or more claims of the '211 Patent include, but are not limited to, at least the Apple Accused Products.

150. The Apple Accused Products directly infringe one or more claims of the '211 Patent. Apple makes, uses, sells, offers for sale, and/or imports, in this District and elsewhere in the United States these devices and thus directly infringes the '211 Patent.

151. Apple has had knowledge and notice of the '211 Patent at least as of August 31, 2014, by virtue of Nokia presenting the '211 Patent to Apple. Apple has been involved in licensing discussions with Nokia regarding Nokia's patent portfolio, which includes the '211 Patent.

152. Apple indirectly infringes the '211 Patent, as provided in 35 U.S.C. § 271(b), by inducing infringement by others, such as Apple's customers and end-users, in this District and elsewhere in the United States. For example, Apple's customers and end-users directly infringe through their use of the inventions claimed in the '211 Patent. Apple induces this direct infringement through its affirmative acts of manufacturing, selling, distributing, and/or otherwise

making available the Apple Accused Products, and providing instructions, documentation, and other information to customers and end users suggesting they use the Apple Accused Products in an infringing manner, including in-store technical support, online technical support, marketing, product manuals, advertisements, online documentation, developer information, and API documentation. As a result of Apple's inducement, Apple's customers and end users use the Apple Accused Products in the way Apple intends and directly infringe the '211 Patent. Apple has performed and continues to perform these affirmative acts with knowledge of the '211 Patent and with the intent, or willful blindness, that the induced acts directly infringe the '211 Patent.

153. Apple also indirectly infringes the '211 Patent, as provided by 35 U.S.C. § 271(c), by contributing to direct infringement committed by others, such as customers and end-users, in this District and elsewhere in the United States. Apple's affirmative acts of selling and offering to sell, in this District and elsewhere in the United States, the Apple Accused Products and causing the Apple Accused Products to be manufactured, used, sold, and offered for sale contribute to Apple's customers and end-users use of the Apple Accused Products, such that the '211 Patent is directly infringed. The accused components within the Apple Accused Products are material to the invention of the '211 Patent, are not staple articles or commodities of commerce, have no substantial non-infringing uses, and are known by Apple to be especially made or especially adapted for use in infringement of the '211 Patent. Apple has performed and continues to perform these affirmative acts with knowledge of the '211 Patent and with intent, or willful blindness, that they cause the direct infringement of the '211 Patent.

154. Apple's infringement of the '211 Patent has damaged and will continue to damage Nokia.

COUNT VII: PATENT INFRINGEMENT OF THE '701 PATENT

155. Nokia incorporates by reference the preceding paragraphs as though fully set forth herein.

156. Apple infringes, contributes to the infringement of, and/or induces infringement of the '701 Patent by making, using, selling, offering for sale, and/or importing into the United States products and/or methods covered by one or more claims of the '701 Patent including, but not limited to, at least the Apple Accused Products. The accused devices that infringe one or more claims of the '701 Patent include, but are not limited to, at least the Apple Accused Products.

157. The Apple Accused Products directly infringe one or more claims of the '701 Patent. Apple makes, uses, sells, offers for sale, and/or imports, in this District and elsewhere in the United States these devices and thus directly infringes the '701 Patent.

158. Apple has had knowledge and notice of the '701 Patent at least as of August 31, 2014, by virtue of Nokia presenting the '701 Patent to Apple. Apple has been involved in licensing discussions with Nokia regarding Nokia's patent portfolio, which includes the '701 Patent.

159. Apple indirectly infringes the '701 Patent, as provided in 35 U.S.C. § 271(b), by inducing infringement by others, such as Apple's customers and end-users, in this District and elsewhere in the United States. For example, Apple's customers and end-users directly infringe through their use of the inventions claimed in the '701 Patent. Apple induces this direct infringement through its affirmative acts of manufacturing, selling, distributing, and/or otherwise making available the Apple Accused Products, and providing instructions, documentation, and other information to customers and end users suggesting they use the Apple Accused Products in

an infringing manner, including in-store technical support, online technical support, marketing, product manuals, advertisements, online documentation, developer information, and API documentation. As a result of Apple's inducement, Apple's customers and end users use the Apple Accused Products in the way Apple intends and directly infringe the '701 Patent. Apple has performed and continues to perform these affirmative acts with knowledge of the '701 Patent and with the intent, or willful blindness, that the induced acts directly infringe the '701 Patent.

160. Apple also indirectly infringes the '701 Patent, as provided by 35 U.S.C. § 271(c), by contributing to direct infringement committed by others, such as customers and end-users, in this District and elsewhere in the United States. Apple's affirmative acts of selling and offering to sell, in this District and elsewhere in the United States, the Apple Accused Products and causing the Apple Accused Products to be manufactured, used, sold, and offered for sale contribute to Apple's customers and end-users use of the Apple Accused Products, such that the '701 Patent is directly infringed. The accused components within the Apple Accused Products are material to the invention of the '701 Patent, are not staple articles or commodities of commerce, have no substantial non-infringing uses, and are known by Apple to be especially made or especially adapted for use in infringement of the '701 Patent. Apple has performed and continues to perform these affirmative acts with knowledge of the '701 Patent and with intent, or willful blindness, that they cause the direct infringement of the '701 Patent.

161. Apple's infringement of the '701 Patent has damaged and will continue to damage Nokia.

COUNT VIII: PATENT INFRINGEMENT OF THE '974 PATENT

162. Nokia incorporates by reference the preceding paragraphs as though fully set forth herein.

163. Apple infringes, contributes to the infringement of, and/or induces infringement of the '974 Patent by making, using, selling, offering for sale, and/or importing into the United States products and/or methods covered by one or more claims of the '974 Patent including, but not limited to, at least the Apple Accused Products. The accused devices that infringe one or more claims of the '974 Patent include, but are not limited to, at least the Apple Accused Products.

164. The Apple Accused Products directly infringe one or more claims of the '974 Patent. Apple makes, uses, sells, offers for sale, and/or imports, in this District and elsewhere in the United States these devices and thus directly infringes the '974 Patent.

165. Apple has knowledge of the '974 Patent. Apple has received actual notice of the '974 Patent as of the date this lawsuit was filed and/or the date this Original Complaint was served upon Apple.

166. Apple indirectly infringes the '974 Patent, as provided in 35 U.S.C. § 271(b), by inducing infringement by others, such as Apple's customers and end-users, in this District and elsewhere in the United States. For example, Apple's customers and end-users directly infringe through their use of the inventions claimed in the '974 Patent. Apple induces this direct infringement through its affirmative acts of manufacturing, selling, distributing, and/or otherwise making available the Apple Accused Products, and providing instructions, documentation, and other information to customers and end users suggesting they use the Apple Accused Products in an infringing manner, including in-store technical support, online technical support, marketing, product manuals, advertisements, online documentation, developer information, and API documentation. As a result of Apple's inducement, Apple's customers and end users use the Apple Accused Products in the way Apple intends and directly infringe the '974 Patent. Apple

has performed and continues to perform these affirmative acts with knowledge of the '974 Patent and with the intent, or willful blindness, that the induced acts directly infringe the '974 Patent.

167. Apple also indirectly infringes the '974 Patent, as provided by 35 U.S.C. § 271(c), by contributing to direct infringement committed by others, such as customers and end-users, in this District and elsewhere in the United States. Apple's affirmative acts of selling and offering to sell, in this District and elsewhere in the United States, the Apple Accused Products and causing the Apple Accused Products to be manufactured, used, sold, and offered for sale contribute to Apple's customers and end-users use of the Apple Accused Products, such that the '974 Patent is directly infringed. The accused components within the Apple Accused Products are material to the invention of the '974 Patent, are not staple articles or commodities of commerce, have no substantial non-infringing uses, and are known by Apple to be especially made or especially adapted for use in infringement of the '974 Patent. Apple has performed and continues to perform these affirmative acts with knowledge of the '974 Patent and with intent, or willful blindness, that they cause the direct infringement of the '974 Patent.

168. Apple's infringement of the '974 Patent has damaged and will continue to damage Nokia.

DAMAGES

169. As a result of Apple's acts of infringement, Nokia has suffered actual and consequential damages. However, Nokia does not yet know the full extent of the infringement and its extent cannot be ascertained except through discovery and special accounting. To the fullest extent permitted by law, Nokia seeks recovery of damages at least for reasonable royalties, unjust enrichment, and benefits received by Apple as a result of using misappropriated

technology. Nokia further seeks any other damages to which Nokia is entitled under law or in equity.

DEMAND FOR JURY TRIAL

170. Nokia hereby demands a jury trial for all issues so triable.

PRAYER FOR RELIEF

WHEREFORE, Nokia respectfully requests that this Court enter judgment in its favor as follows:

A. that Apple infringes the Nokia Patents-in-Suit;

B. that Apple's infringement of the Nokia Patents-in-Suit was willful, and that Apple's continued infringement of these patents is willful;

C. awarding Nokia actual damages in an amount sufficient to compensate Nokia for Apple's infringement of the Nokia Patents-in-Suit until such time as Apple ceases its infringing conduct;

D. awarding enhanced damages pursuant to 35 U.S.C. § 284;

E. awarding Nokia pre-judgment and post-judgment interest to the full extent allowed under the law, as well as its costs;

F. in view of the fact that Apple is an unwilling licensee, entering an injunction in favor of Nokia, precluding Apple and any entities in active concert with it from future acts of infringement;

G. that this is an exceptional case and awarding Nokia its reasonable attorneys' fees pursuant to 35 U.S.C. § 285;

H. ordering an accounting of damages for acts of infringement;

I. awarding Nokia its costs of suit; and

J. awarding such other equitable relief which may be requested and to which Nokia is entitled.

DATED: December 21, 2016

Respectfully submitted,

By: */s/ Theodore Stevenson, III*

Theodore Stevenson, III – Lead
tstevenson@mckoolsmith.com
Nicholas Mathews
nmathews@mckoolsmith.com
McKool Smith, P.C.
300 Crescent Court, Suite 1500
Dallas, Texas 75201
Telephone: (214) 978-4000
Facsimile: (214) 978-4044

Samuel F. Baxter
sbaxter@mckoolsmith.com
McKool Smith, P.C.
104 E. Houston Street, Suite 3000
P.O. Box O
Marshall, Texas 75670
Telephone: (903) 923-9000
Facsimile: (903) 923-9099

Kevin L. Burgess
kburgess@mckoolsmith.com
McKool Smith, P.C.
300 W. Sixth Street, Suite 1700
Austin, Texas 78701
Telephone: (512) 692-8700
Facsimile: (512) 692-8744

John D. Haynes
Patrick J. Flinn
Wesley C. Achey
David S. Frist
Michael C. Deane
Alston & Bird LLP
One Atlantic Center
1201 West Peachtree Street
Suite 4900
Atlanta, GA 30309
Telephone: (404) 881-7000
Facsimile: (404) 881-7777
Email: john.haynes@alston.com
Email: patrick.flinn@alston.com
Email: wes.achey@alston.com
Email: david.frist@alston.com
Email: michael.deane@alston.com

Attorneys for Plaintiffs Nokia Technologies Oy and Alcatel-Lucent USA Inc.